kokoszka FDA CH12

Noa Jeong

Ch 12

```r
rm(list=ls())

library(MASS)
library(fda)
library(ggplot2)
```

12.4

```r
# R.version # R version 4.4.1 (2024-06-14)
```

```r
#install.packages("sp")
#install.packages("RandomFieldsUtils_1.2.5.tar.gz", repos = NULL)
#install.packages('RandomFields_3.3.14.tar.gz',repos = NULL)

#install.packages('CompQuadForm')

library(RandomFields); library(CompQuadForm)
```

- `RandomFields` is used to simulate the functions.
- `CompQuadForm` is used to find the p-values for the norm approach.

A simulation Study

N = 100 with 50 evenly spaced time points over [0,1]

$$X_n(t) = \mu(t) + \epsilon_n(t)$$

with ϵ_n taken to be iid Matern processes with variance=1, scale = 1/4, smoothness = 5/2

mean function μ to be

$$\mu(t) = c_1\sqrt{2}sin((k-1/2)\pi t)$$

First we set an option for RandomFields so that matrices are output.

```r
RFoptions(spConform=FALSE) # makes output a matrix
```

Then we set the parameters of the Matern error

```r
nu<-3/2; var = 1; scale = 1/4
Mat_model<-RMmatern(nu = nu , var = 1 , scale = scale)
N<-10; m<-50;
```

Next we generate the mean function

```
times<-seq(0,1,length=m)
c1<-0; k<-1
mu<-function(x){c1*sqrt(2)*sin((k-1/2)*pi*x)}
mu_vec<-mu(times)
```

Now we simulate the data and tests.

```
reps<-10
TPC3_pval<-numeric(reps)
Tnorm_pval<-numeric(reps)
for(i in 1:reps){
  # Step 1 - Simulate Data
  # Prevent RFsimulate from printing
  capture.output(eps<-RFsimulate(Mat_model,
                                 times,
                                 n=N))
  X_mat<-scale(t(eps),center=-mu_vec,scale=FALSE)
  X.f<-Data2fd(times,t(X_mat))
  # Step 2 - Estimate parameters
  muhat<-mean.fd(X.f)
  X.pca<-pca.fd(X.f,nharm=20)
  lambda<-X.pca$values
  scores<-X.pca$scores
  v<-X.pca$harmonics
  # Step 3 - Compute tests and p-values
  # PCA test with 3 PCs
  TPC3 <- N*sum(inprod(v[1:3],muhat)^2/lambda[1:3])
  TPC3_pval[i]<-pchisq(TPC3,3,lower.tail=FALSE)
  # Norm test
  Tnorm<-N*sum(inprod(v[lambda[1:20]>0],muhat)^2)
  Tnorm_pval[i]<-imhof(Tnorm,lambda[lambda[1:20]>0])[[1]]}
```

`RFsimulate` : Simualtion of Random Fields simulate **unconditional** random fields

12.8 Chapter 12 Problems

Q. 12.3

Recall the simulation scheme given in **Problem 1.5.** For the three ν scenarios, increase N to 1000.

```r
rm(list=ls())
library(fda)
library(ggplot2)
#install.packages('tidyverse')
library(tidyverse)
```

(a) Plot the estimates for the first 10 eigenvalues and include confidence intervals. Include everything in one plot if you can (but use no more than three plots).

(?RMmatern)

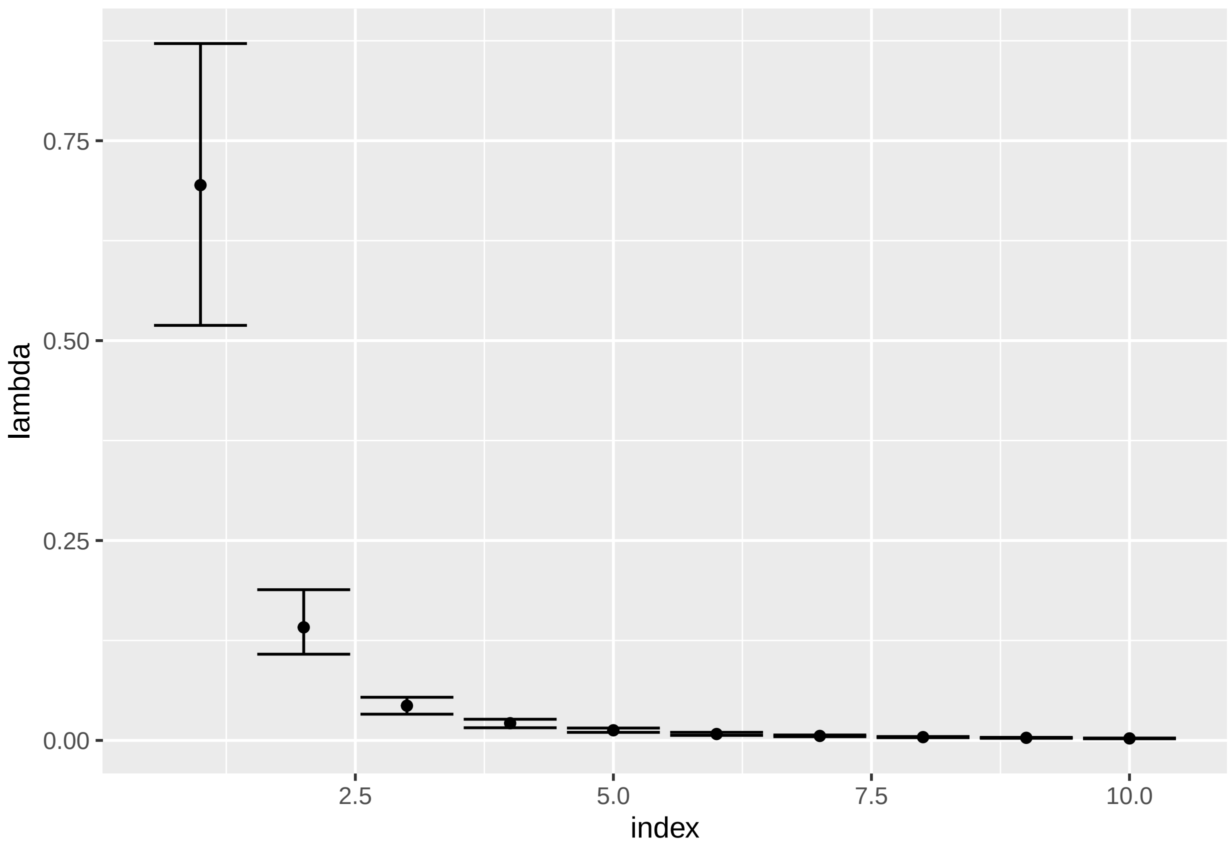
- RMmatern : stationary isotropic covariance model belonging to matern family.
- RFsimulate : Simulate functional data based on specific covariance function

(b) How many of the CI's did not include 0? Suppose you used that to determine the number of FPC's; how does that compare to the explained variance approach?

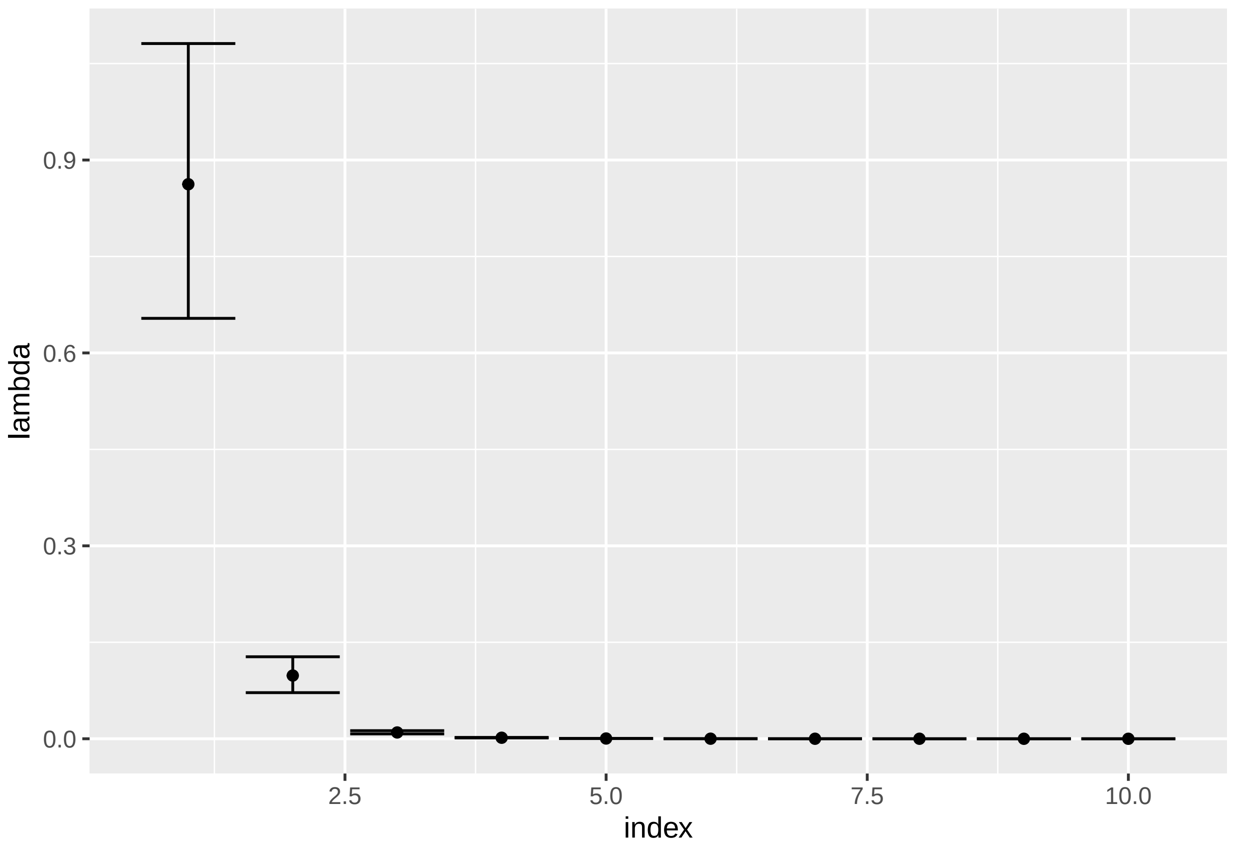
```r
RFoptions(spConform=FALSE)
```

```r
set.seed(1)

# set parameters
nu_values <- c(0.5,2,4); sigma2 <- 1; rho <- 1
N <- 1000
m <- 50 # time points

# generate mean function
time_points <- seq(0, 1, length.out = m)
c1 <- 0; k <- 1
mu <- function(x){c1*sqrt(2)*sin((k-1/2)*pi*x)}
mu_vec <- mu(time_points)


n_obs <- 100
n_simul <- 100
lambda_boot <- matrix(NA, nrow=n_simul, ncol=10)

# simualte data
for(i in 1:3){ # for each nu values
  # epsilon_n taken to b iid Matern process
  Mat_model <- RMmatern(nu_values[i],
                        var = sigma2,
                        scale = rho)
  capture.output(eps <- RFsimulate(Mat_model, # cov function
                                   time_points,
                                   n=N)) # number of function to simulate
  X_mat <- scale(t(eps), center=-mu_vec, scale=F)

  for(n in 1:n_simul){
    idx <- sample(1:1000,
                  size=n_obs,
                  replace=T) # bootstrap
    boot_data <- X_mat[idx, ]
    X.f_boot <- Data2fd(time_points,t(boot_data)) # n_obs x 50
    X.pca_boot <- pca.fd(X.f_boot, nharm=10)
    lambda_boot[n,] <- X.pca_boot$values[1:10] # store lambda for each simulation
  }

  lambda_mean <- apply(lambda_boot,2,mean) # 10 means
  lambda_upper <- apply(lambda_boot, 2, quantile, probs = 0.975)
  lambda_lower <- apply(lambda_boot, 2, quantile, probs = 0.025)

  # Create data frame for plotting
  df <- data.frame(index = 1:10,
                   lambda = lambda_mean,
                   lower_ci = lambda_lower,
                   upper_ci = lambda_upper)
  print(df)

  # (a) Plot
  p <- ggplot(df, aes(x = index, y = lambda)) +
    geom_point() +
    geom_errorbar(aes(ymin = lower_ci, ymax = upper_ci))
  plot(p)

  # (b)
  eps <- 1e-5
  print("Number of lower bound which didn't include zero")
  print(sum(df$lower_ci > eps))

}
```

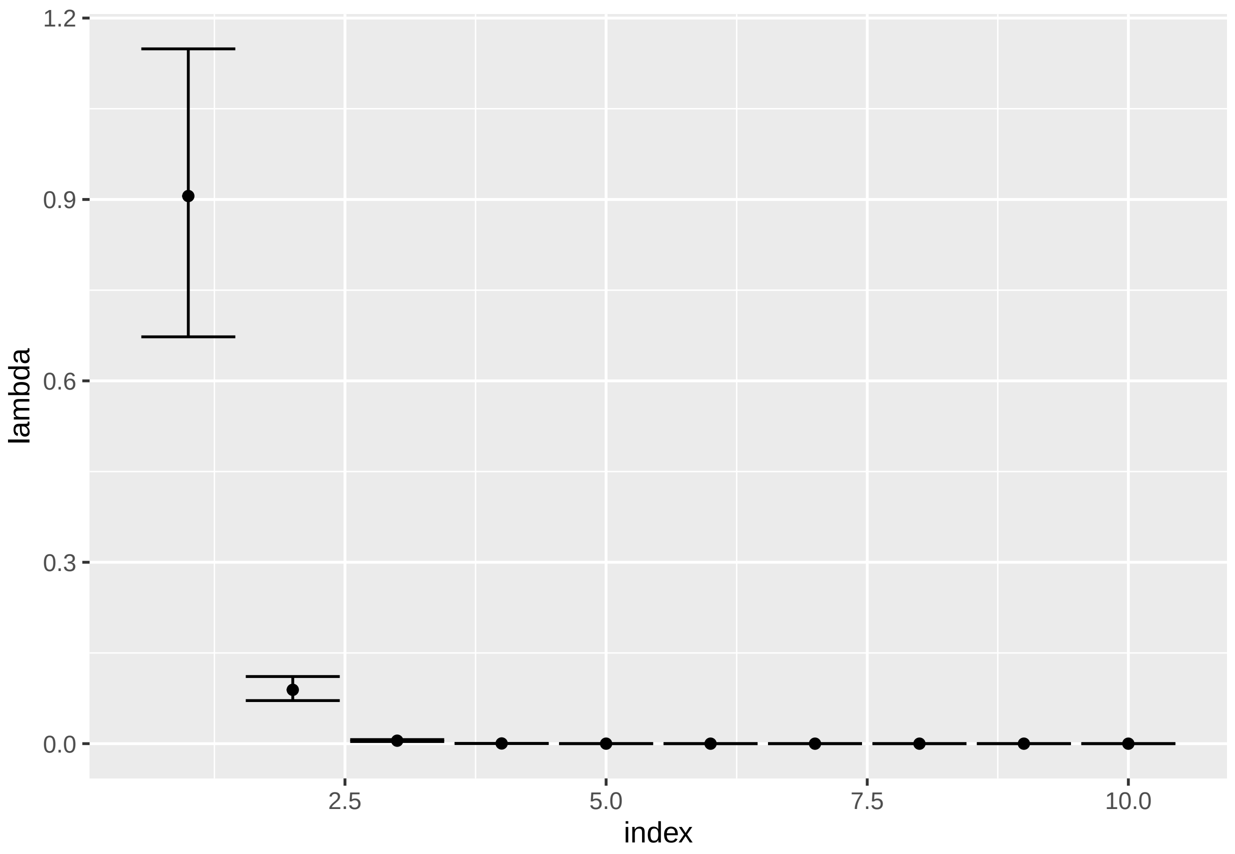
```
##    index      lambda    lower_ci    upper_ci
## 1      1 0.694487872 0.519083225 0.871546145
## 2      2 0.141429359 0.107831543 0.188470906
## 3      3 0.043418284 0.032809395 0.053959372
## 4      4 0.021506069 0.015833181 0.026460216
## 5      5 0.012676885 0.010099589 0.015469613
## 6      6 0.008005701 0.006333439 0.010140677
## 7      7 0.005610553 0.004644423 0.006780902
## 8      8 0.004089491 0.003433266 0.004845912
## 9      9 0.003250265 0.002668572 0.003821691
## 10    10 0.002505968 0.002076948 0.002969684
```



```
## [1] "Number of lower bound which didn't include zero"
## [1] 10
##    index       lambda     lower_ci     upper_ci
## 1      1 8.623459e-01 6.537747e-01 1.081124e+00
## 2      2 9.831152e-02 7.170210e-02 1.275237e-01
## 3      3 9.766825e-03 7.549945e-03 1.259205e-02
## 4      4 1.614062e-03 1.269398e-03 2.081995e-03
## 5      5 3.900461e-04 3.002004e-04 4.958374e-04
## 6      6 1.201683e-04 8.621141e-05 1.517107e-04
## 7      7 4.639688e-05 3.363293e-05 5.952475e-05
## 8      8 2.081278e-05 1.497916e-05 2.725017e-05
## 9      9 9.635724e-06 7.072579e-06 1.336549e-05
## 10    10 5.714163e-06 4.402666e-06 7.359772e-06
```



```
## [1] "Number of lower bound which didn't include zero"
## [1] 8
##    index       lambda     lower_ci     upper_ci
## 1      1 9.056405e-01 6.728329e-01 1.149074e+00
## 2      2 8.902727e-02 7.121938e-02 1.110310e-01
## 3      3 4.933502e-03 3.531364e-03 6.718481e-03
## 4      4 3.490900e-04 2.577908e-04 4.392147e-04
## 5      5 3.049370e-05 2.352960e-05 4.084469e-05
## 6      6 3.775631e-06 2.939439e-06 4.708874e-06
## 7      7 6.061465e-07 4.632660e-07 7.729334e-07
## 8      8 1.162896e-07 8.274968e-08 1.590170e-07
## 9      9 3.138775e-08 2.440129e-08 4.010714e-08
## 10    10 9.828241e-09 7.731122e-09 1.242722e-08
```



```
## [1] "Number of lower bound which didn't include zero"
## [1] 5
```