kokoszka FDA CH2

Noa Jeong

2. Further topics in exploratory analysis of functional data

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

```
library(fda)
```

2.1 Derivatives

$x_n(t)$: observation, k times continously differentiable

$B_m(t)$: particular basis, basis function must have at least k derivatives

e.g Fourier basis : infinitely differentiable. // B-spline : higher order than cubic if one derivative

- kth order derivative function

$$x_n^{(k)}(t) \approx \sum_{m=1}^{M} c_{mn} B^{(k)}(t)$$

Example 2.1.1

The Matern process is a Gaussian process.

$\nu = 5/2$ and other σ, ρ is equal to 1.

$$\begin{aligned} c(t,s) &= (\sqrt{5}|t-s|)^{5/2} K_\nu(\sqrt{5}|t-s|) \\ &= (1+\sqrt{5}|t-s| + (5/3)|t-s|^2)\exp(-\sqrt{5}|t-s|) \end{aligned}$$

Note. Bessel function approx

$$K_{5/2}(x) = \sqrt{\frac{\pi}{2x}}\exp(-x)(1+\frac{3}{x}+\frac{3}{x^2})$$
$$x = \sqrt{5}|t-s|$$

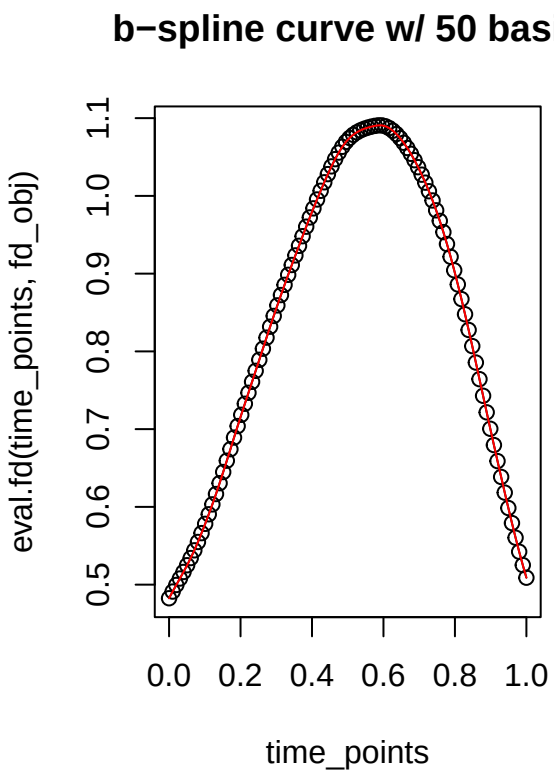
simulate above Matern process on 100 evenly spaced time points on [0,1]

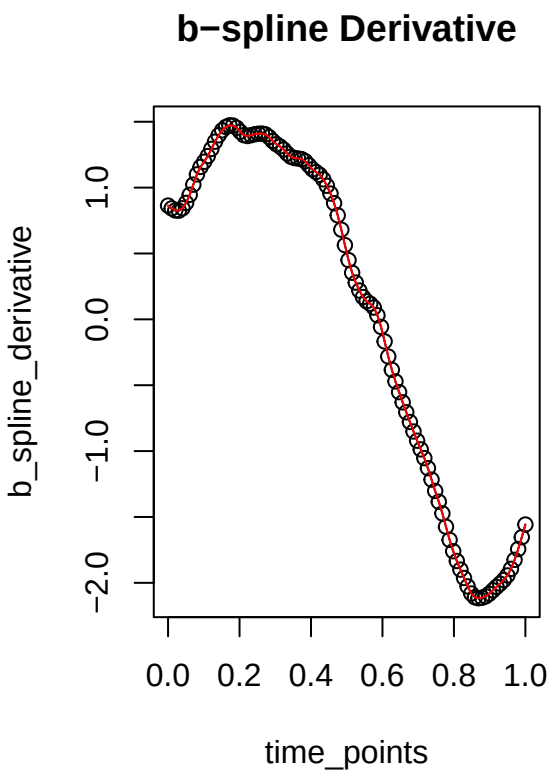
- Figure 2.1

```
matern_cov <- function(t,s){
  (1+sqrt(5)*abs(t-s)+(5/3)*abs(t-s)^2)*exp(-sqrt(5)*abs(t-s))
  } # each element in covariance

time_points <- seq(0, 1, length.out = 100)
```

```r
# create 100x100 matern_cov matrix
create_cov_mat <- function(time_points){
  n <- length(time_points)
  cov_mat <- matrix(0,n,n)
  for(i in 1:n){
    for(j in 1:n){
      cov_mat[i,j] <- matern_cov(time_points[i],time_points[j])
    }
  }
  return(cov_mat)
}

# simulation functions
simul <- function(n_sim, time_points){
  n <- length(time_points)
  cov_mat <- create_cov_mat(time_points)
  simulation <- mvrnorm(n_sim, mu=rep(0,n), Sigma = cov_mat)
  return(simulation)
}
```

```r
set.seed(2)
simulation_result <- simul(1, time_points)

basis <- create.bspline.basis(rangeval=c(0,1), nbasis=50)
fd_obj <- smooth.basis(time_points, simulation_result, basis)$fd

fd_deriv <- deriv.fd(fd_obj)
b_spline_derivative <- eval.fd(time_points, fd_deriv)

par(mfrow=c(1,2))
plot(time_points, eval.fd(time_points,fd_obj), type = 'o', main = "b-spline curve w/ 50 basis",)
lines(time_points,eval.fd(time_points,fd_obj),col='red') # red line in left panel indicates a b-splines

plot(time_points, b_spline_derivative, type = 'o',
     main = "b-spline Derivative",)
lines(time_points,b_spline_derivative,col='red') # the red line in the right panel is its derivative.
```


b−spline curve w/ 50 basis
eval.fd(time_points, fd_obj)
0.5
0.6
0.7
0.8
0.9
1.0
1.1
0.0
0.2
0.4
0.6
0.8
1.0
time_points


b−spline Derivative
b_spline_derivative
−2.0
−1.0
0.0
1.0
0.0
0.2
0.4
0.6
0.8
1.0
time_points

2.2 Penalized Smoothing

imposes penalty on functions that are too "wiggly"
raw data curves : a substantial level of noise
functional objects with large M will inherit this variability
wiggles are due to measurement error or random variability

- Curve Smoothing

observe values y_j at points $t_j \in [T_1, T_2]$
(t_j, y_j) : observed data points
smooth curve $x(t), t \in [T_1, T_2]$.

$y_j = x(t_j) + \epsilon_j$

goal of smoothing is to eliminate the ϵ_j and obtain estimate of $x(t)$

$$x(t) \approx x_K(t) = \sum_{k=1}^{K} c_k B_k(t)$$

$K >> M$ where M is number of points t_j

- Linear operator (linear combination of m derivatives of x)

$$L(x)(t) = a_0(t)x(t) + a_1(t)x^{(1)}(t) + ... + a_m(t)x^{(m)}(t)$$

e.g second derivative $L(x)(t) = x^{(2)}(t)$

Find the coefficient c_K which minimize the penalized sum of squares

$$PSS_\lambda(c_1, c_2, ..., c_K) = \sum_j (y_j - x_K(t)j)^2 + \lambda \int_{T_1}^{T_2} [L(x_K)(t)]^2 dt$$

λ : smoothing parameter
tuning parameter, generalized cross-validation(GCV) is employed.
suppose data $Y \in \mathbb{R}^N$

$$\hat{Y} = \mathbf{H}_\lambda Y$$

- GCV score

minimize λ which minimize GCV

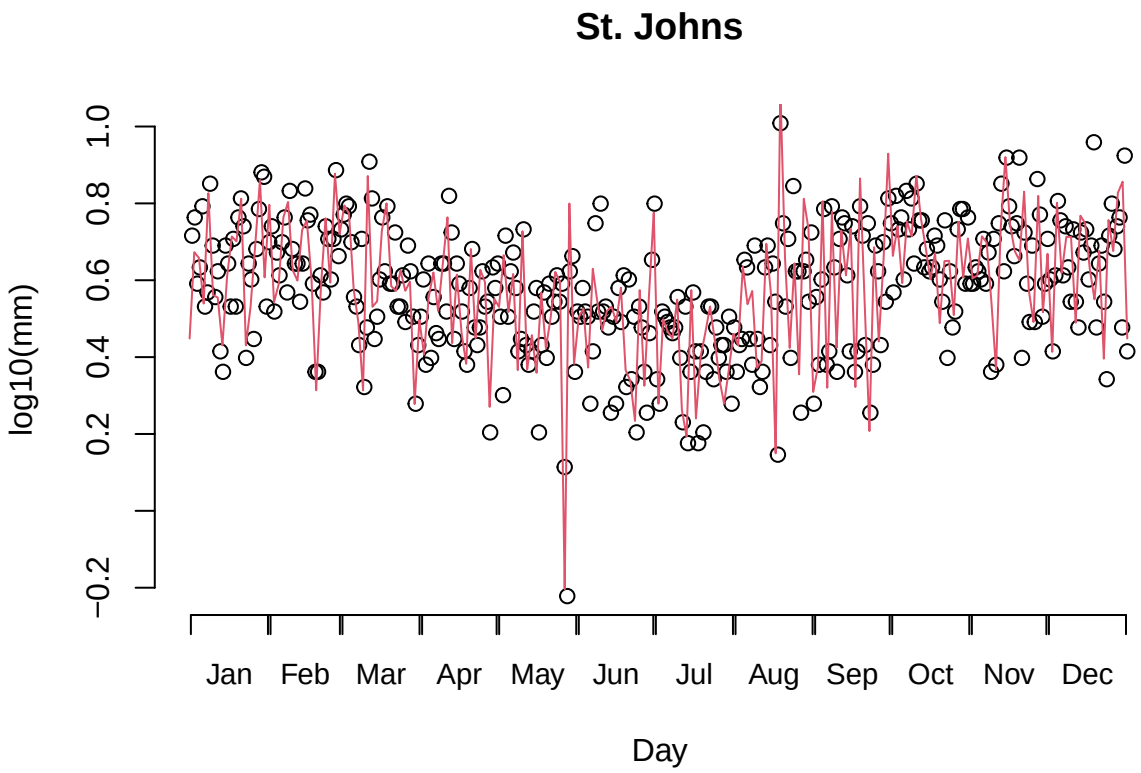
$$\frac{N^{-1}\Sigma_{n=1}^{N}(Y_n - \hat{Y}_n)^2}{(1 - N^{-1}trace(\mathbf{H}_\lambda))^2}$$

Example 2.2.2

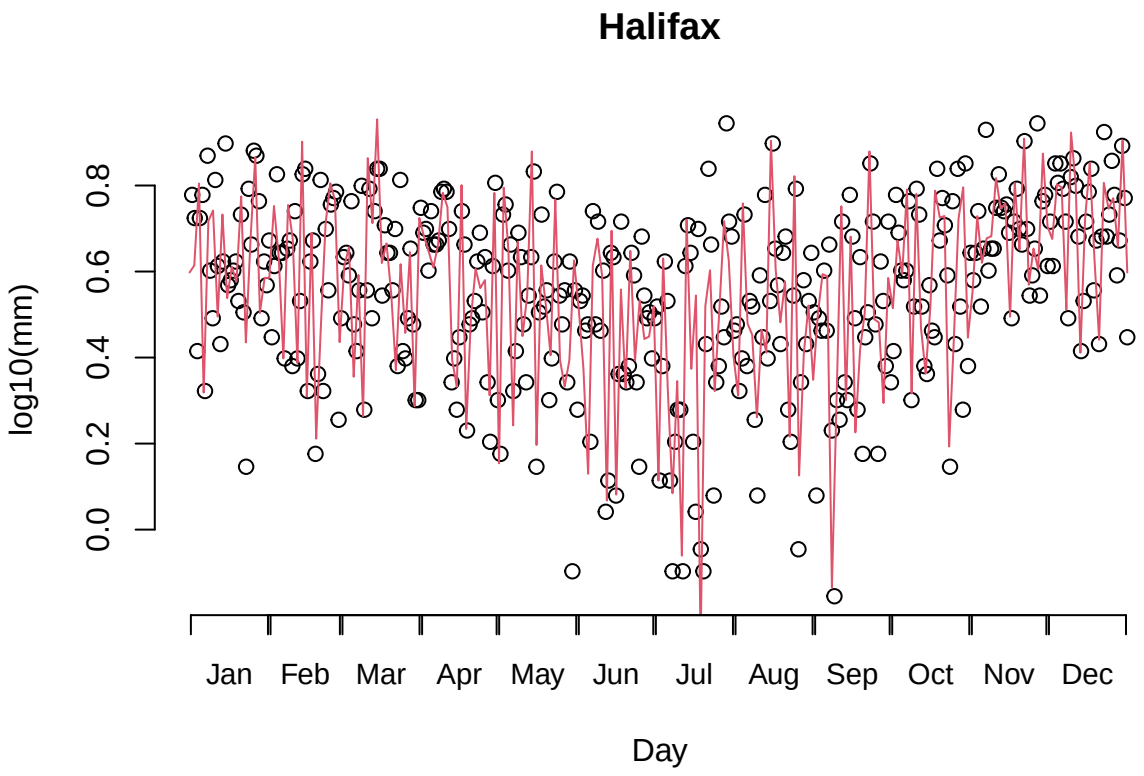
log precipitation data : collect across several stations in Canada

```
# set up a saturated basis : as many basis
# functions as observation

nbasis = 365
yearRng = c(0,365)
daybasis = create.fourier.basis(yearRng, nbasis)
logprecav = CanadianWeather$dailyAv[,,'log10precip']
dayprecfd <- with(CanadianWeather,
                  smooth.basis(day.5, # argvals
```

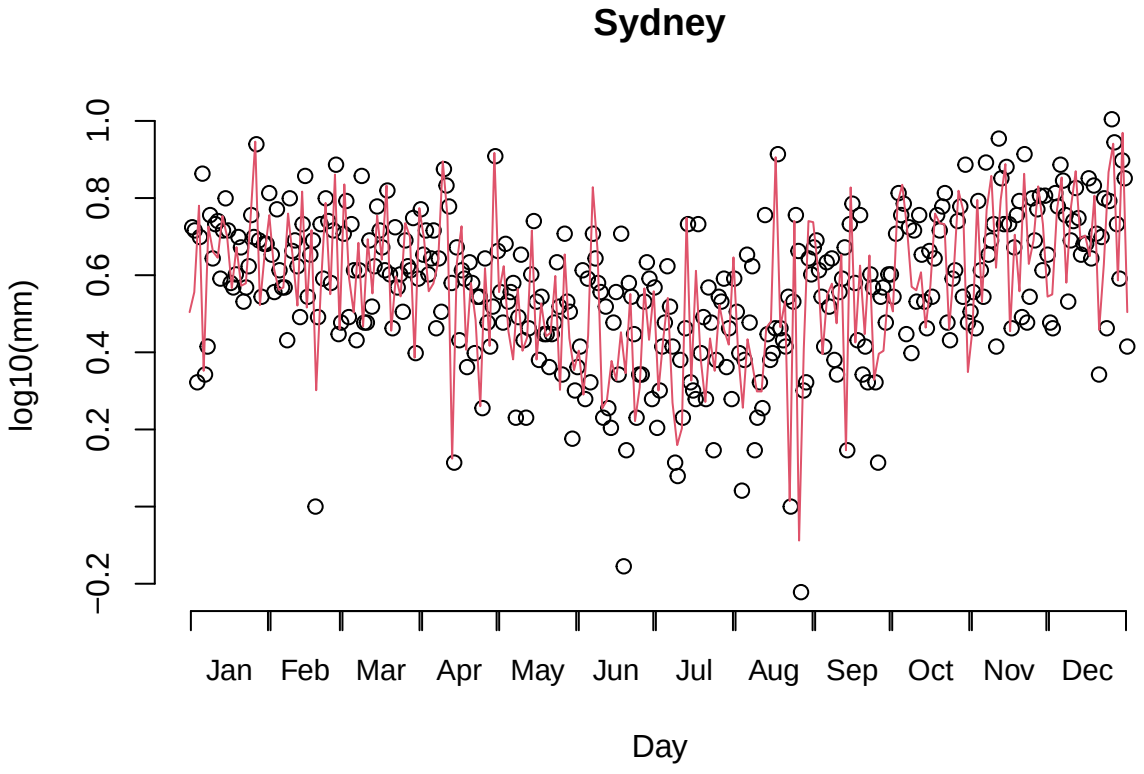
```
                              logprecav, # y
                              daybasis, # fdParaobj(functional basis object)
                              fdnames=list("Day",
                                           "Station",
                                           "log10(mm)"))$fd)
for(i in 1:5){ # first 5 location out of datasets
  plot(logprecav[,i],axes=FALSE,xlab='Day',ylab='log10(mm)',
       main=CanadianWeather$place[i])
  lines(dayprecfd[i],col=2)
  axisIntervals(1)
  axis(2)
  readline('Press <return to continue')
} # esc to stop
```



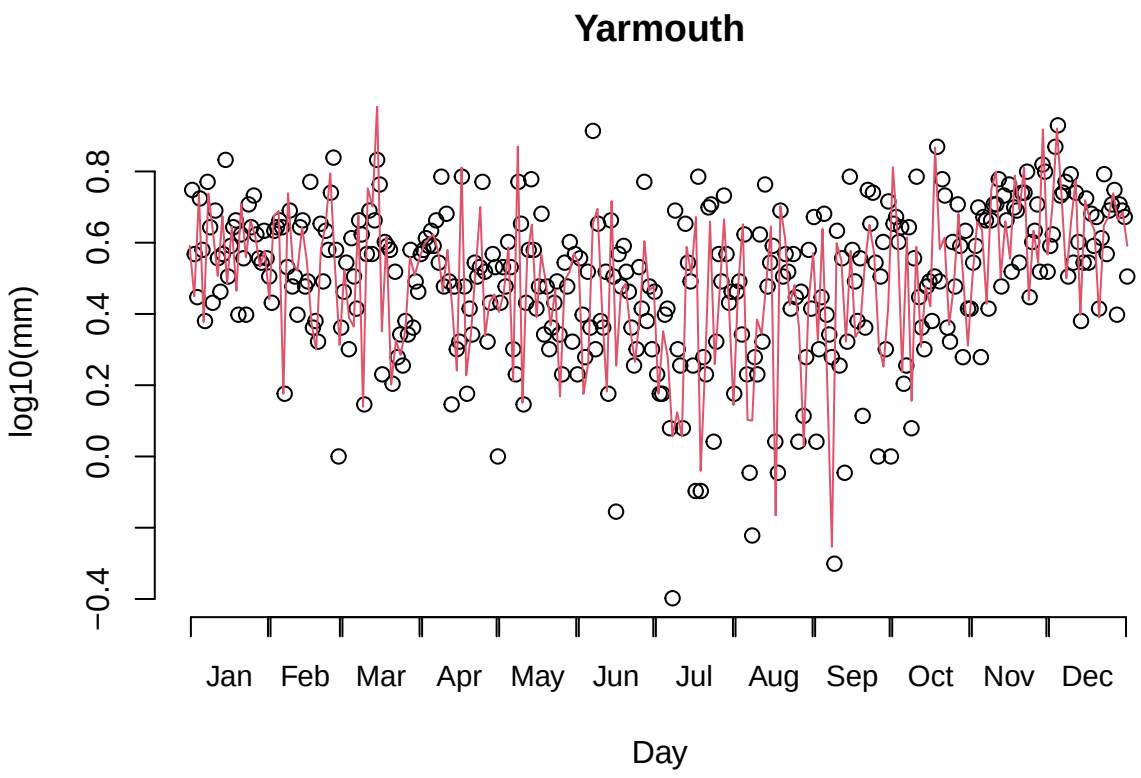

```
## Press <return to continue
```



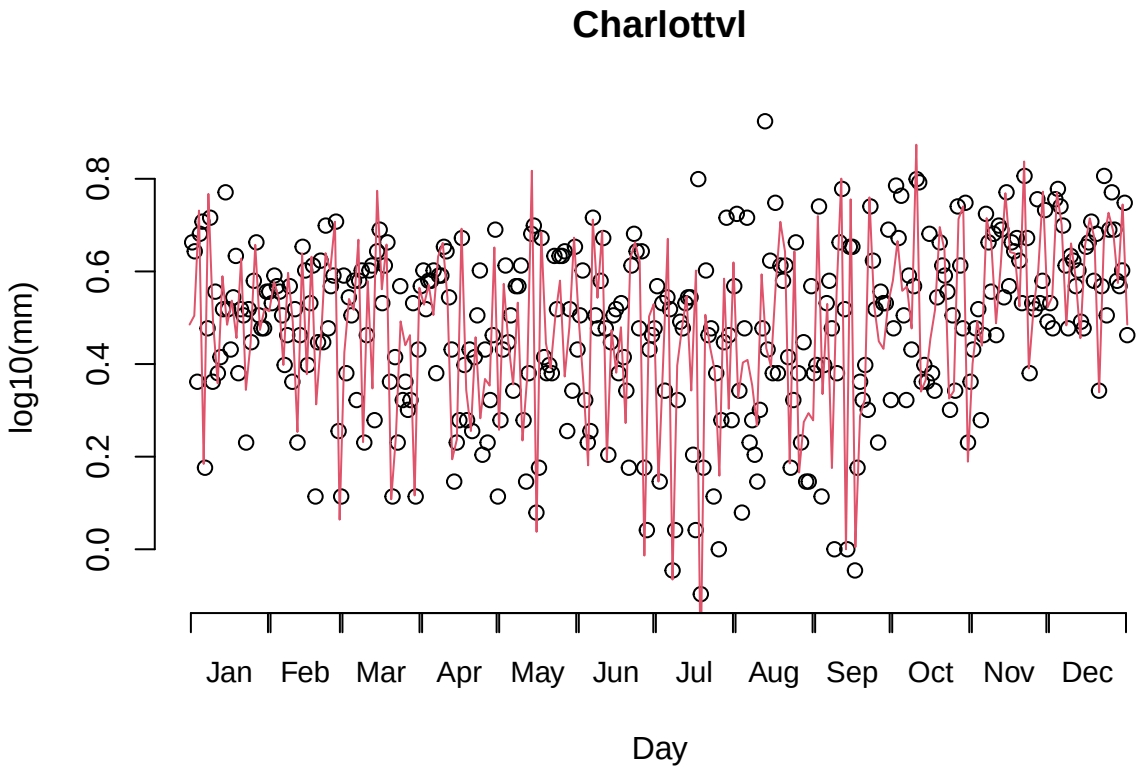

```
## Press <return to continue
```




```
## Press <return to continue
```




```
## Press <return to continue
```




```
## Press <return to continue
```

```
?smooth.basis
```

`smooth.basis` : Construct a functional data object by smoothing data using a roughness penalty. $x_K(t)$

observations with a set of smooth curves, each defined by expansion in term of basis functions.
(1) a set of observed noisy values, (2) a set of argument values associated with these data, and (3) a specification of the basis function system used to define the curves

harmonic accleartion operator is defined by

$$L(x)(t) = \omega^2 x^{(1)}(t) + x^{(3)}(t) \qquad \omega = \frac{2\pi}{T}$$

- smoothing using penalty imposed by the harmonic accelartion operator

```
Lcoef = c(0, (2*pi/diff(yearRng))^2, 0,1)
harmaccelLfd= vec2Lfd(Lcoef, yearRng)
```

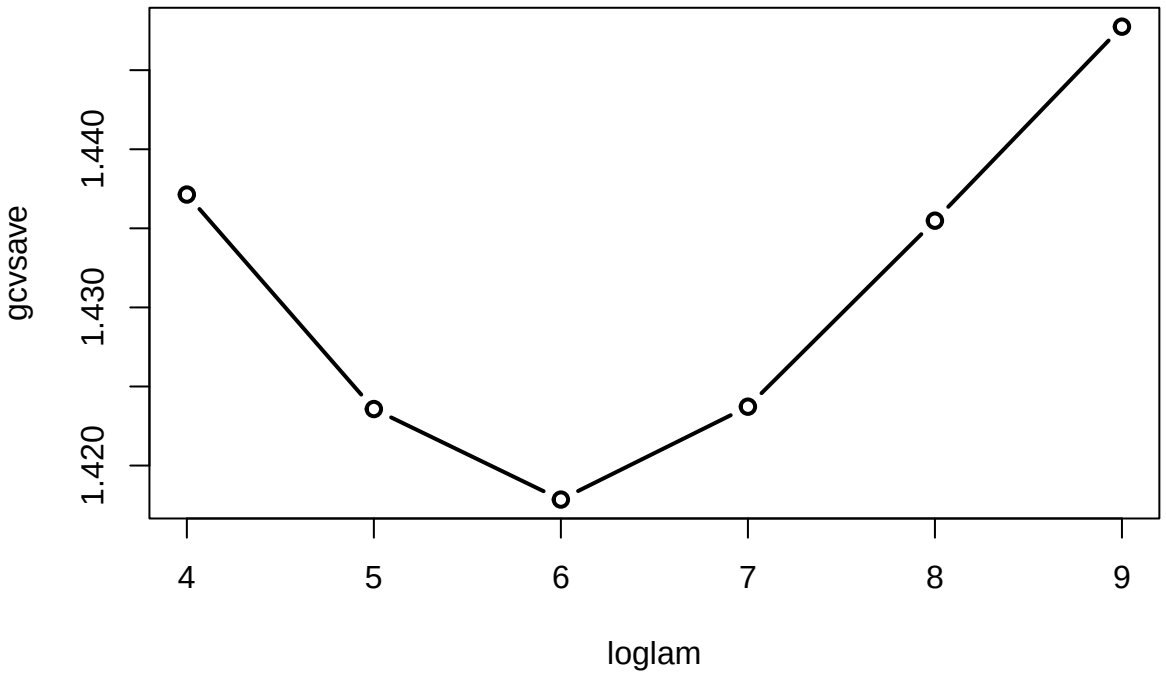
- find λ which minimizes the GCV
  $\lambda = 10^i, 4 \leq i \leq 9$

```
loglam = 4:9
nlam = length(loglam)
dfsave = rep(NA,nlam)
names(dfsave) = loglam
gcvsave = dfsave

for(ilam in 1:nlam){
  cat(paste('log10 lambda =', loglam[ilam], '\n'))
  lambda = 10^loglam[ilam]
  # ?fdPar
  # fdPar : Define a functional param object
  fdParaobj = fdPar(daybasis, # fdobj : functional basis object
                    harmaccelLfd, # linear differential opeartor object L(x)(t)
                    lambda)

  smoothlist = smooth.basis(day.5, logprecav, fdParaobj) # smoothing function
  dfsave[ilam] = smoothlist$df
  gcvsave[ilam] = sum(smoothlist$gcv)

}
```

```
## log10 lambda = 4
## log10 lambda = 5
## log10 lambda = 6
## log10 lambda = 7
## log10 lambda = 8
## log10 lambda = 9
```

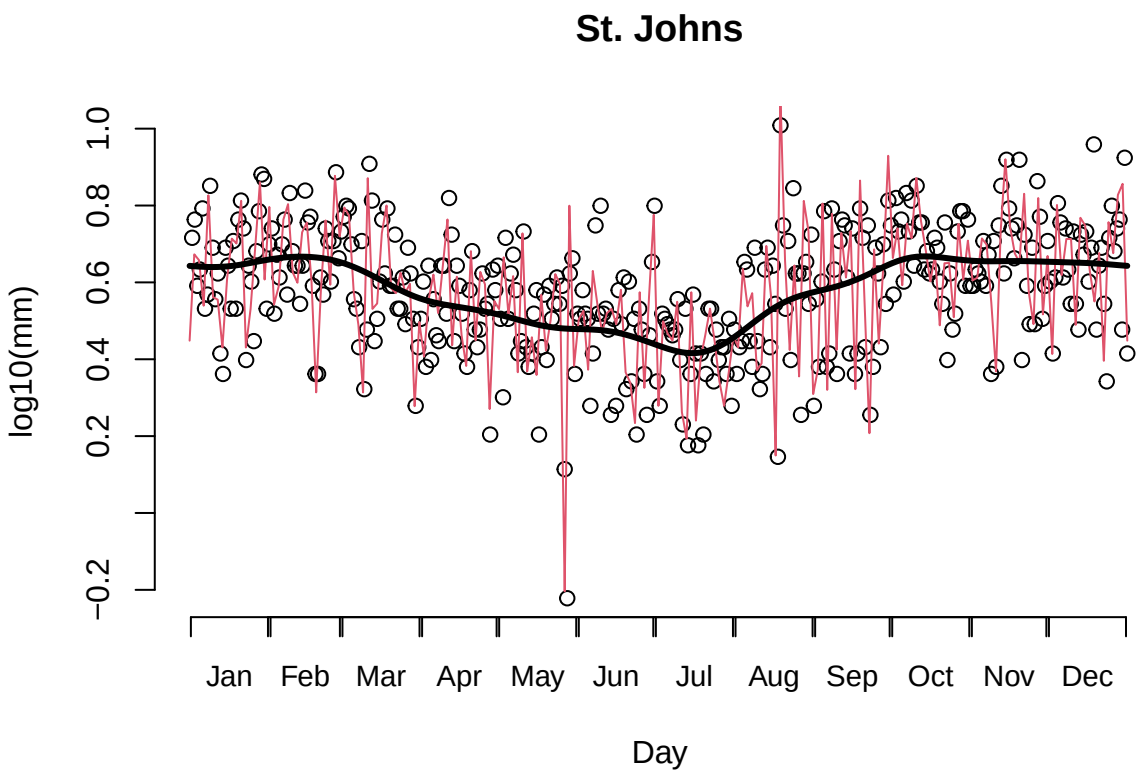
```
## Warning in smooth.basis1(argvals, y, fdParobj, wtvec = wtvec, fdnames =
## fdnames, : lambda reduced to 389800616.78291 to prevent overflow
```

```
plot(loglam,gcvsave,type='b',lwd=2)
```

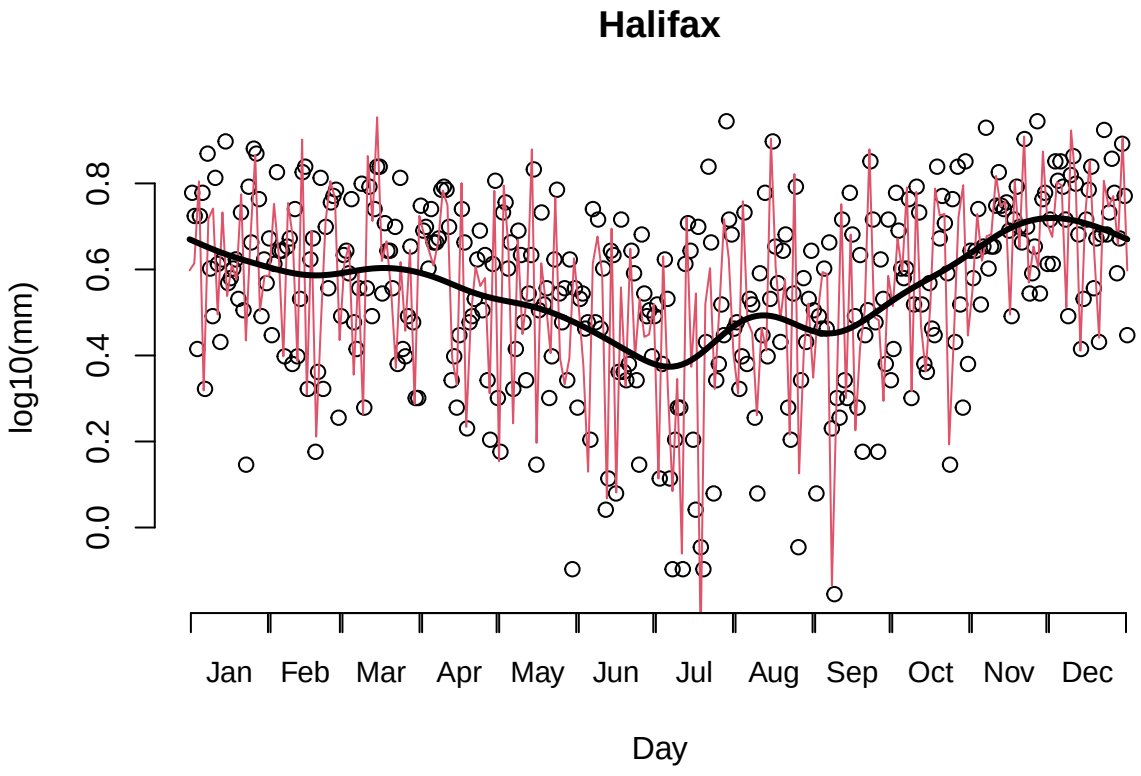


-> $\lambda = 10^6$ minimizes the GCV

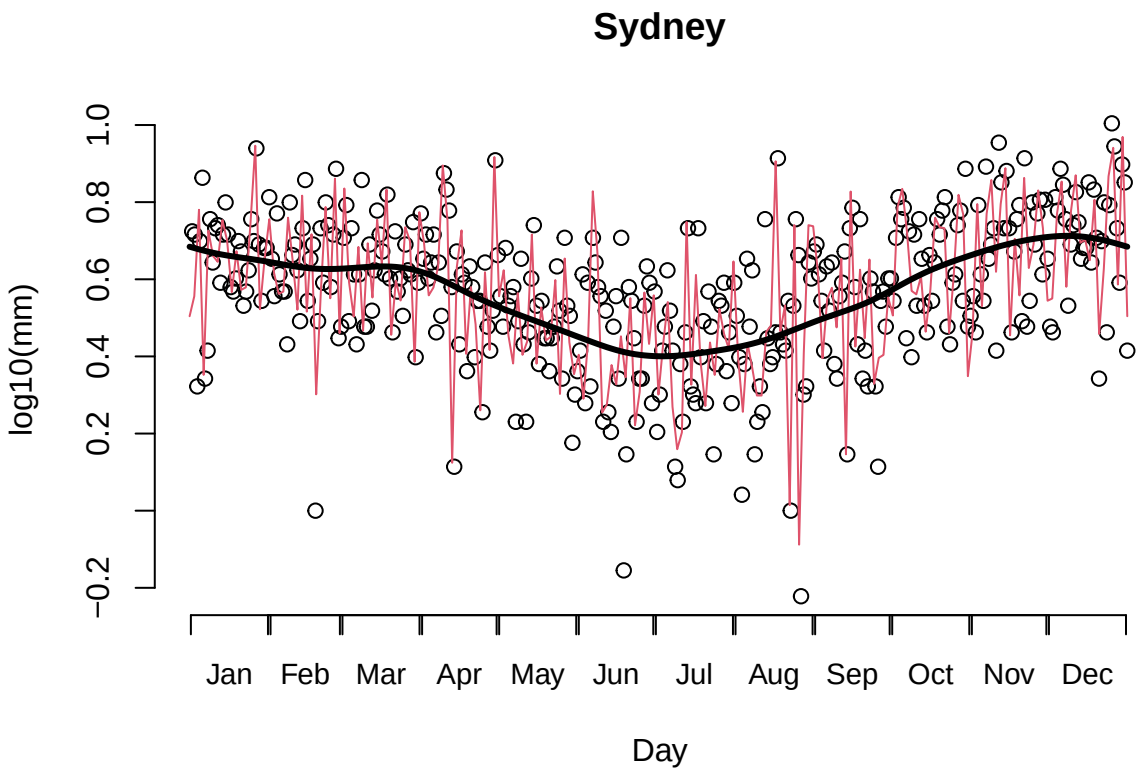
```
lambda = 1e6
fdParaobj = fdPar(daybasis, # basis function
                  harmaccelLfd, # L(x)(t), differential operator
                  lambda)
logprec.fit = smooth.basis(day.5, logprecav, fdParaobj) # x_K(t) smoothing function
logprec.fd = logprec.fit$fd
fdnames=list('Day (July 1 to June 30)',
             'Weather Station'=CanadianWeather$place,
             'Log10 Precipitation(mm)')
logprec.fd$fdnames = fdnames

# plot smoothed curves for the first five locations
for(i in 1:5){
  plot(logprecav[,i],axes=FALSE,xlab='Day',ylab='log10(mm)',
       main=CanadianWeather$place[i])
  lines(dayprecfd[i],col=2)
  axisIntervals(1)
  axis(2)
  lines(logprec.fd[i], lwd=3) # smoothing function
  readline('Press <return to continue')
} # esc to stop
```



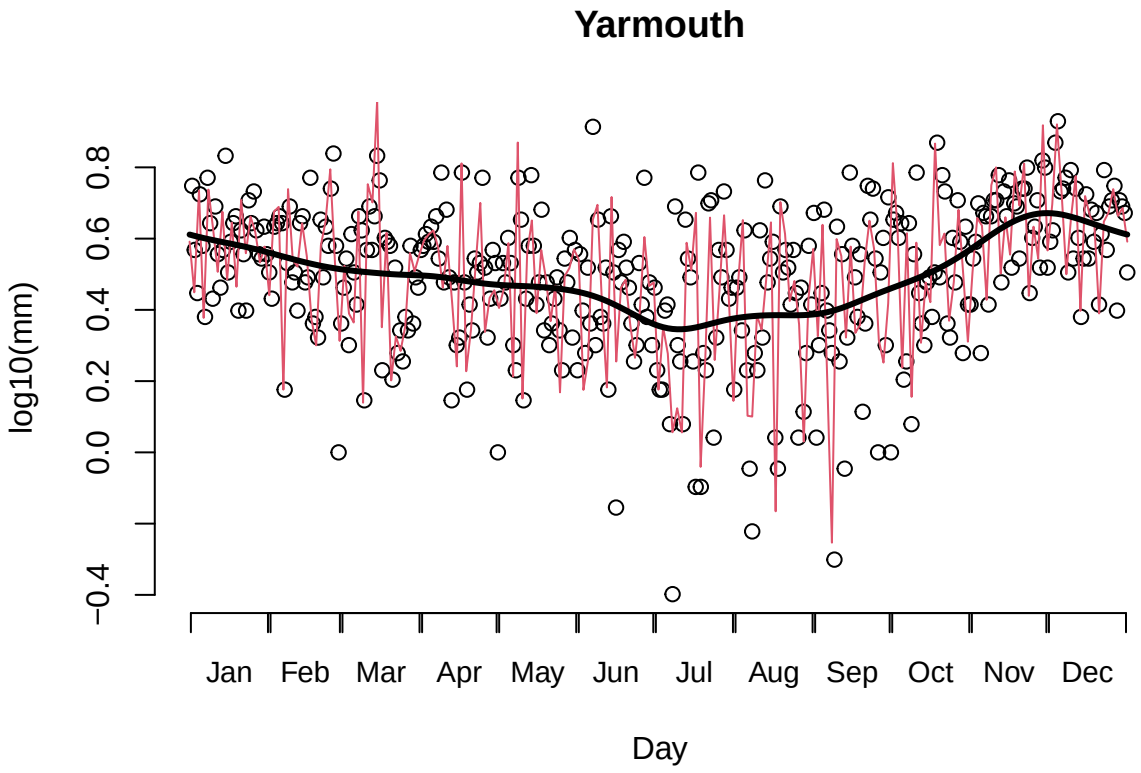

```
## Press <return to continue
```



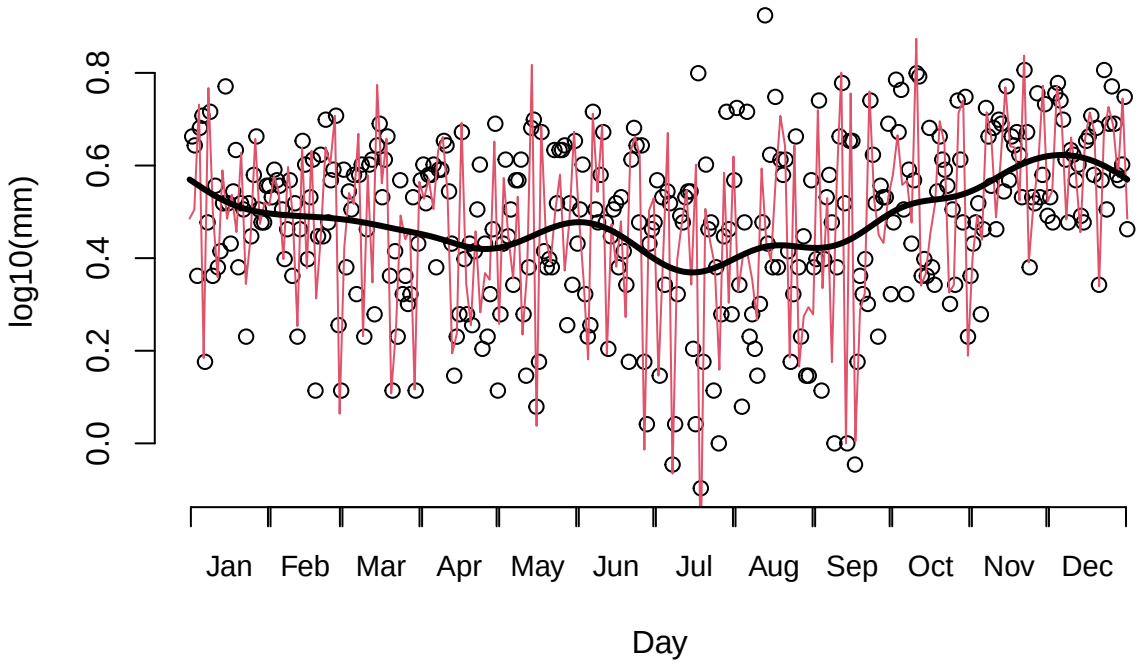

```
## Press <return to continue
```




```
## Press <return to continue
```




```
## Press <return to continue
```




```
## Press <return to continue
```

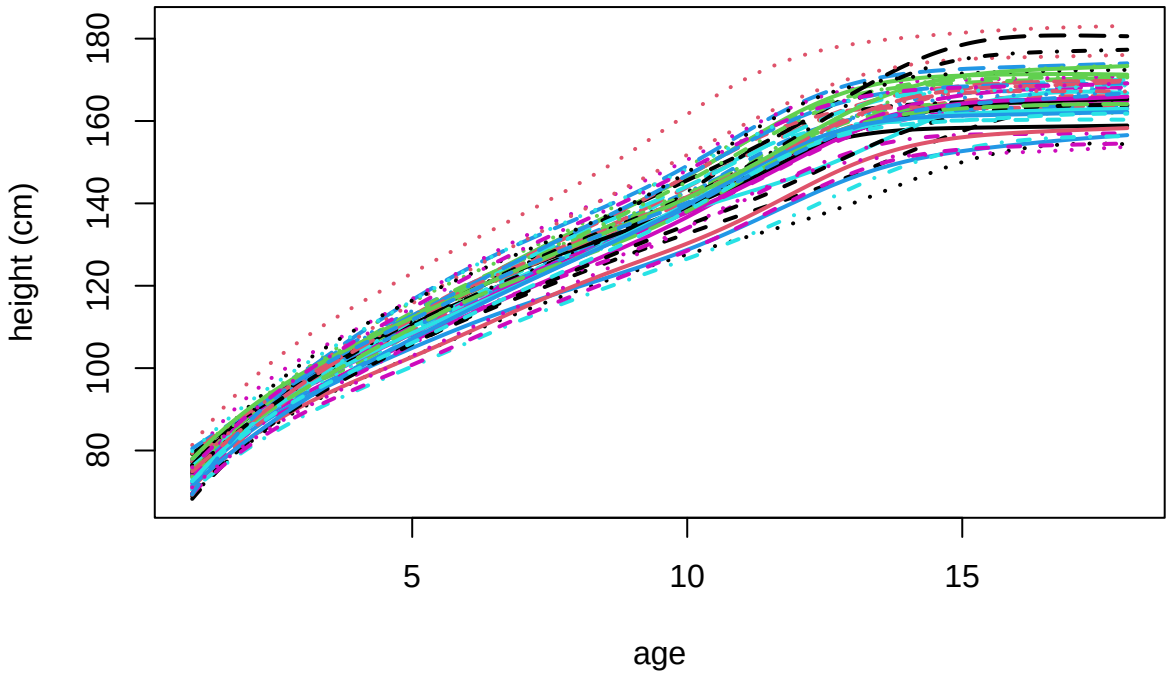
2.3 Curve alignmnet

align the curves in the sample and then compute statistics(mean, covariance, FPC..)

1. amplitude variation : random curve-to-curve variation
2. phase variation : constitutes possible shifting of the curves

- growth curves of 54 girls

```
#data(growth)
age = growth$age

heightBasis = create.bspline.basis(c(1,18), # rangeVal
                                   35, # nbasis
                                   6, # norder
                                   age)
# functional parameter
heightPar = fdPar(heightBasis, # fdobj w/ basis object
                  3, # Lfdobj L(x)(t)
                  10^(-0.5)) # lambda
# smooth function
heightSmooth = smooth.basis(age, # argvals 3~18
                            growth$hgtf, # y - height
                            heightPar) # fdParaobj
plot(heightSmooth,lwd=2,xlab='age',ylab='height (cm)')
```




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

individual growth curves differ by level(amplitude), by timing of certain landmark events

- acceleration curves of 54 girls with the mean funciton.

second derivatives of the growth function .

`deriv.fd`: Compute a Derivative of a Functional Data Object Description

```
accelUnreg = deriv.fd(heightSmooth$fd, # functional data object
                      2)
```

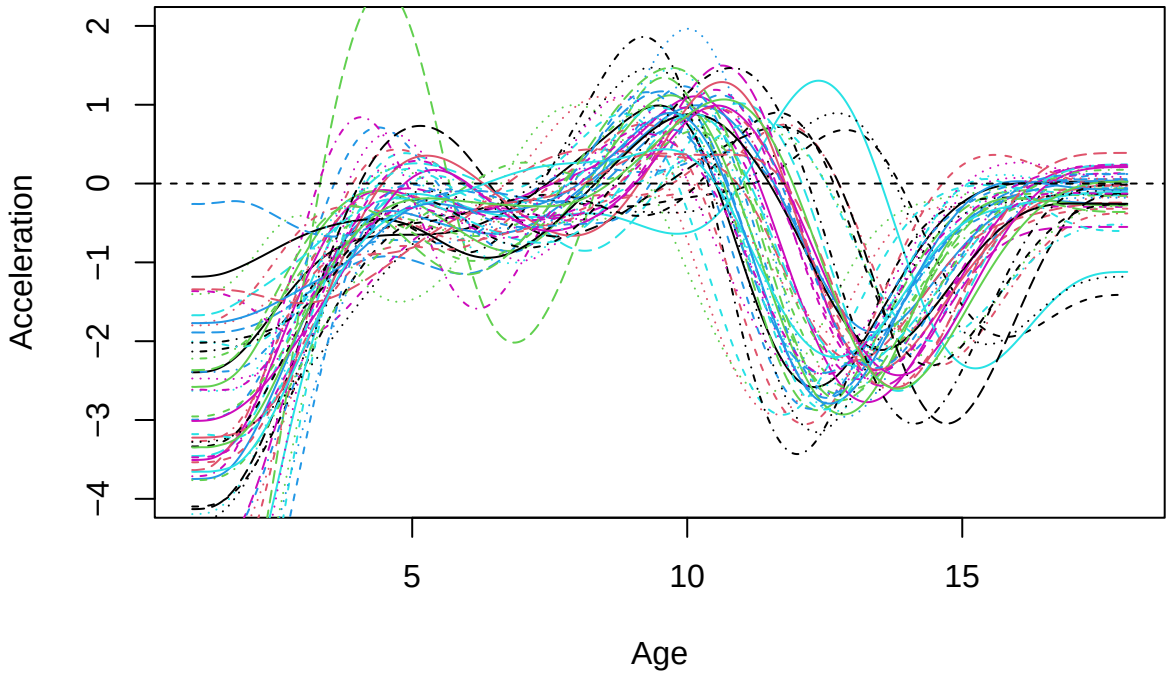
```
plot(accelUnreg[,1], lwd=1,xlab='Age',ylab='Acceleration',
     ylim=c(-4,2))
```

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

```
mean.accelUnreg = mean(accelUnreg) # mean function
```

```
## Warning in mean.default(accelUnreg): argument is not numeric or logical:
## returning NA
```

```
lines(mean.accelUnreg, lwd=8, col='black')
```



both amplitude and phase variation.

1. landmark registration

Approach to align curves
time warping function $h_n(t)$: stretches or shrinks time

t = physical time
$h_n(t)$ = subject specific time.

assume that observed curves $X_n(t)$ are

$$X_n(t) = X_n^*(h_n(t))$$
$$X_n^*(t) = X_n(h_n^{-1}(t))$$

where $X_n^*(t)$ are properly aligned curves

Estimate the $h_n(t)$ using landmark registration.

e.g. growth curve of 54 girls
t_n : physical time when individuals n grows the fastest
t_a : average time of the fastest growth.
$h_n(t_n) = t_a$ <=> all individuals time t_n get mapped to the same value t_a.

Thus, every individuals have three points in the plane $(0,0),(t_n,t_a),(T,T)$.
X_n : original curve
$X_n^{reg}(t) = X_n(h_n^{-1}(t))$

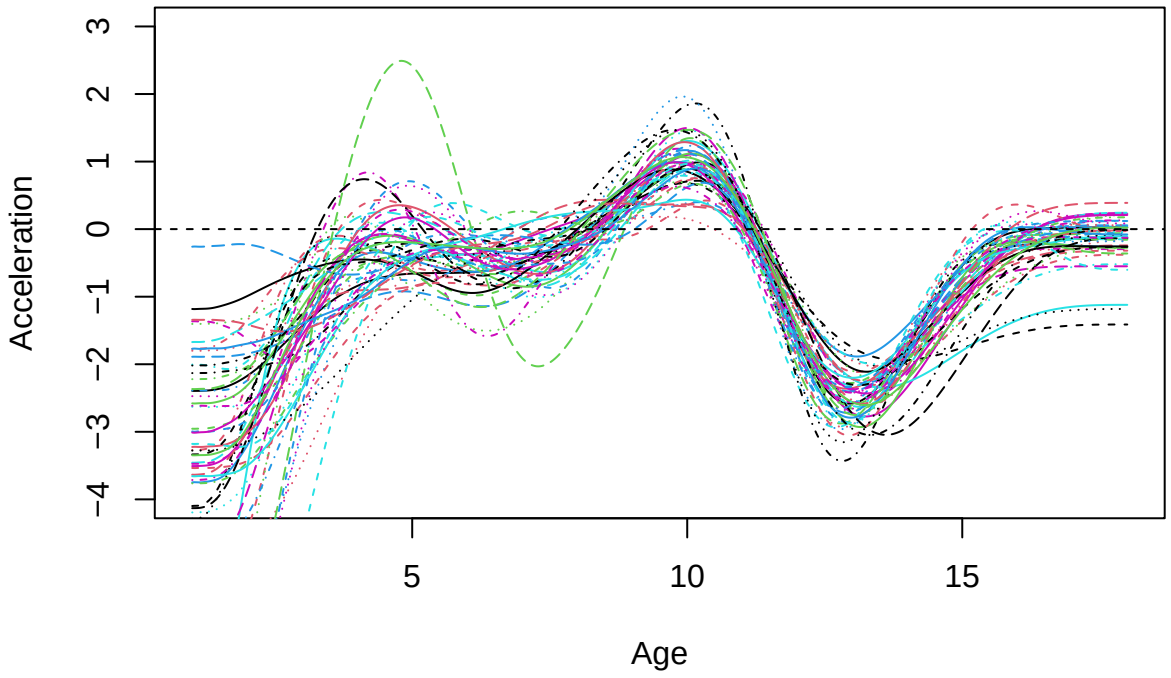
if the number of curves to be registered is large -> `locator()`.

2. continuous registration no discrete landmarks are used.

more automated alignment procedures -> `register.fd` : calculates increasing warping function h_n, so that registered curves become close to their mean

```
regList = register.fd(yfd=accelUnreg)
```

```
accelReg = regList$regfd
#$regfd extracts the registered functions
plot(accelReg, xlab='Age',ylab='Acceleration',ylim=c(-4,3))
```




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

- Total sample variance of unregistered curves X_i

decomposition into variance due to amplitude and phase variability.

$$MSE_{total} = \frac{1}{N}\sum_{n=1}^{N}\int [X_n(t) - \bar{X}(t)]^2 dt$$
$$= MSE_{amp} + MSE_{pha}$$

```
warpFunctions=regList$warpfd
# $warpfd extracts the warping functions
APList = AmpPhaseDecomp(xfd=accelUnreg,
                        yfd=accelReg,
                        hfd=warpFunctions)

APList$RSQR # extract proportion of the total variation due to phas variation : MSpha / MSE total
```

```
## [1] 0.3900564
```

```
APList$MS.amp
```

```
## [1] 5.917822
```

```
APList$MS.pha
```

```
## [1] 3.784422
```

2.5 Chapter 2 Problem

```
rm(list=ls())
library(fda)
```

2.2

Download the R package fds and use the data set FedYieldcurve, which contains the monthly Federal Reserve interest rates, cf. Problem 1.2.

```
library(fds)
data("FedYieldcurve")
str(FedYieldcurve)
```

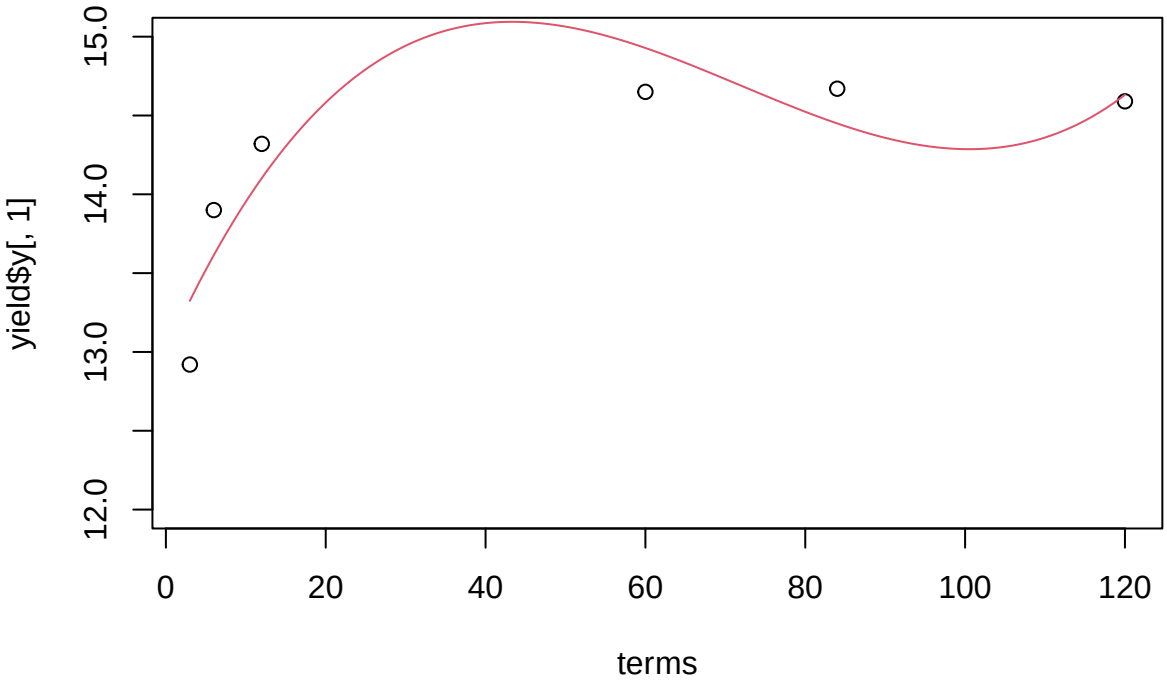
```
## List of 5
##  $ x    : num [1:6] 3 6 12 60 84 120
##  $ y    : num [1:6, 1:330] 12.9 13.9 14.3 14.7 14.7 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:6] "3" "6" "12" "60" ...
##   .. ..$ : chr [1:330] "1" "2" "3" "4" ...
##  $ time : Time-Series [1:330] from 1 to 330: 1 2 3 4 5 6 7 8 9 10 ...
##  $ xname: chr "Maturity term"
##  $ yname: chr "Interest rate"
##  - attr(*, "class")= chr [1:2] "fts" "fds"
```

(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$x # 3   6  12  60  84 120
yield$y[,1] # 1982 Jan values
```

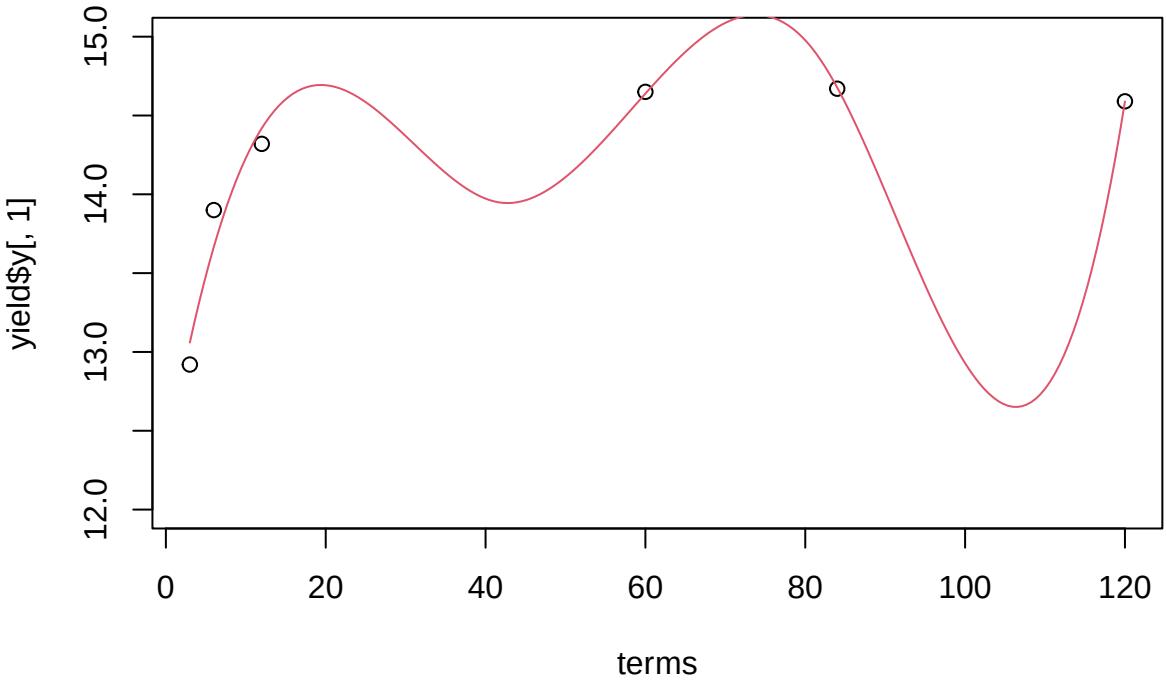
```
##     3     6    12    60    84   120
## 12.92 13.90 14.32 14.65 14.67 14.59
```

```
b_spline_basis <- create.bspline.basis(range(3,120),nbasis = 4)
y_fd <- smooth.basis(terms,
                     yield$y[,1],
                     b_spline_basis) # basis

plot(terms,yield$y[,1],ylim=c(12,15))
lines(y_fd,col=2)
```



(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 $\lambda = 1$, 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.

```
b_spline_basis <- create.bspline.basis(range(3,120),nbasis = 6)

# lambda
lambda <- 1

# second derivative
norder = norder(y_fd)
Lfdobj <- int2Lfd(max(0, norder-2))

# ?fdPar # bspline !
fdParaobj <- fdPar(b_spline_basis, # fdobject
                   Lfdobj, # linear differential operator object L(x)(t)
                   lambda)
new_y_fd <- smooth.basis(terms,
                         yield$y[,1],
                         fdParaobj) # smoothing function

plot(terms,yield$y[,1],ylim=c(12,15))
lines(new_y_fd,col=2)
```

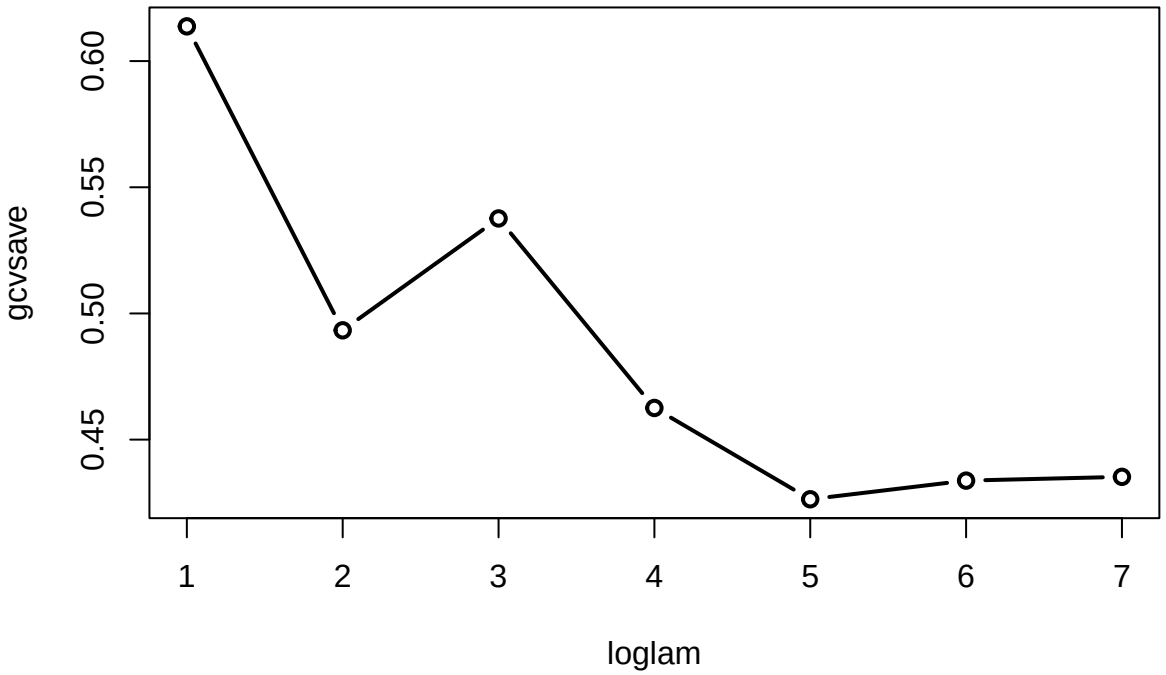


(c) Repeat part (b) with several other smoothing parameters. Which gives the most informative smooth curve?

```
loglam = 1:7
nlam = length(loglam)
dfsave = rep(NA,nlam)
names(dfsave) = loglam
gcvsave = dfsave

for(ilam in 1:nlam){
  cat(paste('log10 lambda =', loglam[ilam], '\n'))
  lambda = 10^loglam[ilam]
  fdParaobj = fdPar(b_spline_basis, # functional basis object
                    Lfdobj, # linear differential opeartor object L(x)(t)
                    lambda)

  smoothlist = smooth.basis(terms,yield$y[,1], fdParaobj) # smoothing function
  dfsave[ilam] = smoothlist$df
  gcvsave[ilam] = sum(smoothlist$gcv)

}
```

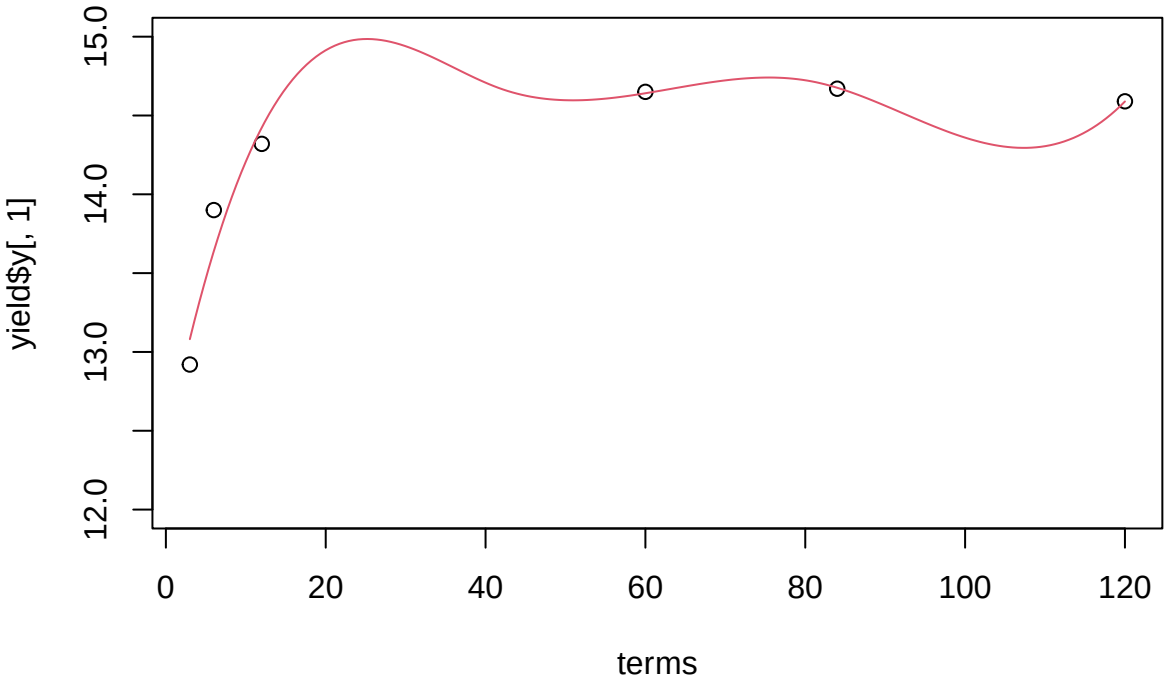
```
## log10 lambda = 1
## log10 lambda = 2
## log10 lambda = 3
## log10 lambda = 4
## log10 lambda = 5
## log10 lambda = 6
## log10 lambda = 7
```

```
plot(loglam,gcvsave,type='b',lwd=2)
```



- $\lambda = 5$

```
lambda <- 5
fdParaobj <- fdPar(b_spline_basis, # fdobject
                   Lfdobj,
                   lambda)
new_y_fd <- smooth.basis(terms,
                         yield$y[,1],
                         fdParaobj) # smoothing function

plot(terms,yield$y[,1],ylim=c(12,15))
lines(new_y_fd,col=2)
```


yield$y[, 1]
15.0
14.0
13.0
12.0
0
20
40
60
80
100
120
terms

2.4

Consider the DTI data in Section 1.5.

```
rm(list=ls())
library(refund)
library(fda)
data(DTI)
str(DTI)
```

```
## 'data.frame':    382 obs. of  9 variables:
##  $ ID        : num  1001 1002 1003 1004 1005 ...
##  $ visit     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ visit.time: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Nscans    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ case      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ sex       : Factor w/ 2 levels "male","female": 2 2 1 1 1 1 1 1 1 1 ...
##  $ pasat     : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ cca       : num [1:382, 1:93] 0.491 0.472 0.502 0.402 0.402 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:382] "1001_1" "1002_1" "1003_1" "1004_1" ...
##   .. ..$ : chr [1:93] "cca_1" "cca_2" "cca_3" "cca_4" ...
##  $ rcst      : num [1:382, 1:55] 0.257 NaN NaN 0.508 NaN ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:382] "1001_1" "1002_1" "1003_1" "1004_1" ...
##   .. ..$ : chr [1:55] "rcst_1" "rcst_2" "rcst_3" "rcst_4" ...
```

```
Y <- DTI$cca
Y <- Y[-c(126,130,131,125,319,321),] # missing values
N <- dim(Y)[1]; M <- dim(Y)[2]
# N : number of patients(observations/functions) 376
# M : number of locations(time/spatial observed point)  93
```

(a) Use **penalized smoothing** with 100 basis functions and GCV to convert the data to functional objects. Plot the data and the mean function. Comment on any differences with the direct splines expansion plotted in Figure 1.12.

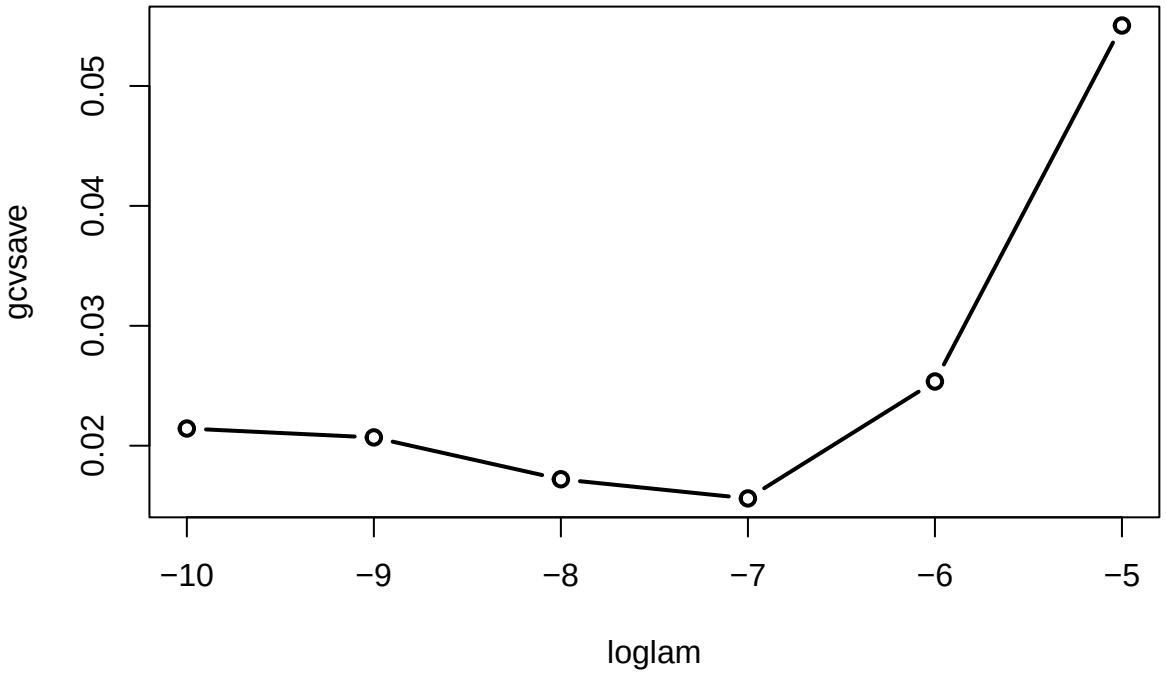
```
argvals <- seq(0,1,length=M) # length 93
data_basis <- create.bspline.basis(c(0,1), # rangeval
                                   nbasis=100)
# bspline : linear differential operator
Lfdobj <- int2Lfd(2)
```

```
# use GCV to find best lambda
loglam = -10:-5
nlam = length(loglam)
dfsave = rep(NA,nlam)
names(dfsave) = loglam
gcvsave = dfsave
for(ilam in 1:nlam){
  cat(paste('log10 lambda =', loglam[ilam], '\n'))
  lambda = 10^loglam[ilam]
  # fdPar : Define a functional param object
  fdParaobj = fdPar(data_basis, # functional basis object
                    Lfdobj, # L(x)(t)
                    lambda)
```

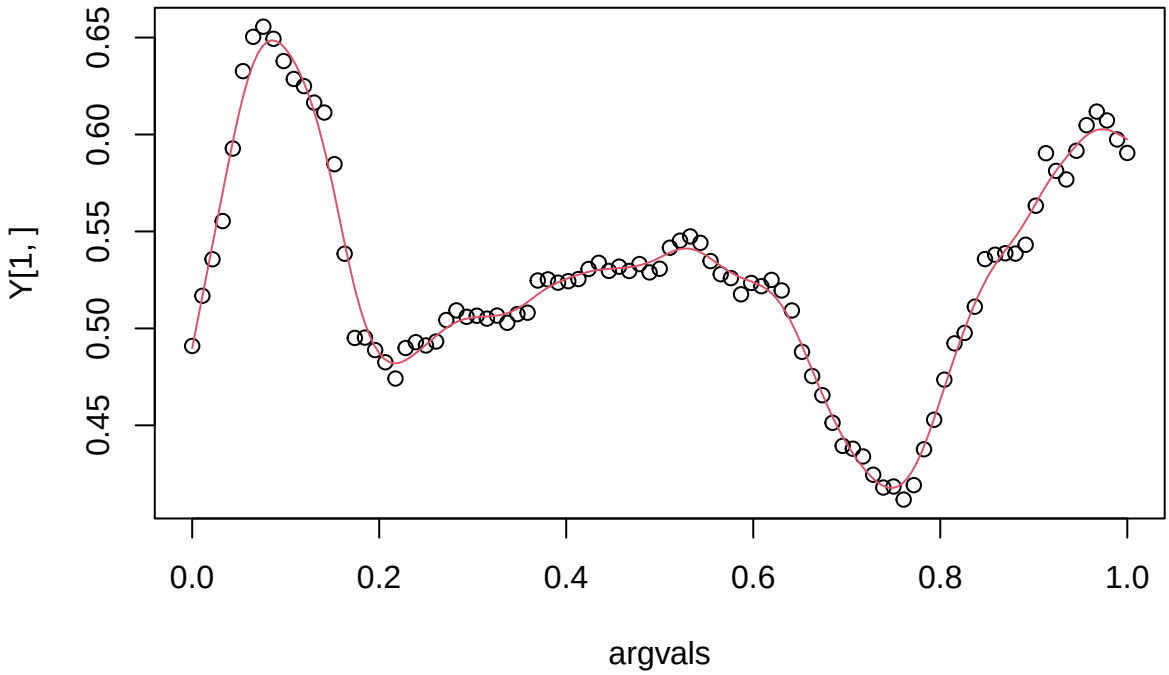
```
  smoothlist = smooth.basis(argvals,
                            t(Y),
                            fdParaobj) # smoothing function
  dfsave[ilam] = smoothlist$df
  gcvsave[ilam] = sum(smoothlist$gcv)
}
```

```
## log10 lambda = -10
## log10 lambda = -9
## log10 lambda = -8
## log10 lambda = -7
## log10 lambda = -6
## log10 lambda = -5
```

```
plot(loglam,gcvsave,type='b',lwd=2)
```



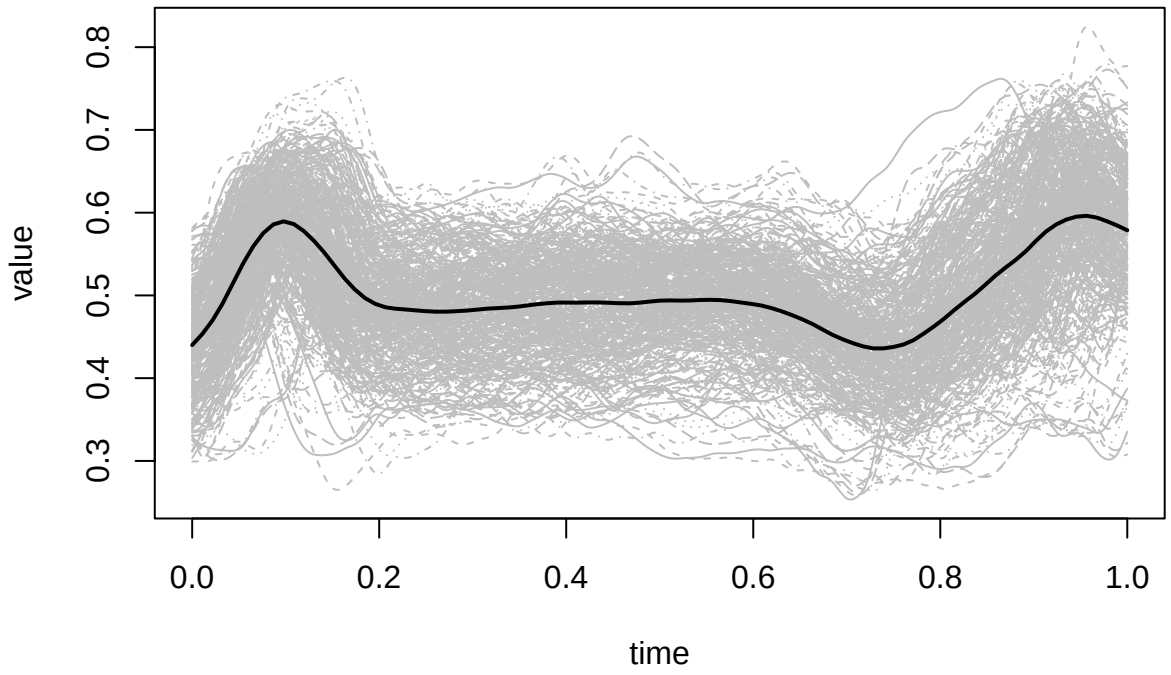

```
plot(argvals,Y[1,])
lines(smoothlist$fd[1],col=2)
```




```
data_par <- fdPar(data_basis,Lfdobj,10^(-7))
data_smooth <- smooth.basis(argvals,
                            t(Y),
                            data_par)
plot(data_smooth,col='gray')
```

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

```
# Evaluate the smoothed data
smoothed_values <- eval.fd(argvals, data_smooth$fd)
# Compute the mean function
lines(argvals,rowMeans(smoothed_values),lwd=2)
```



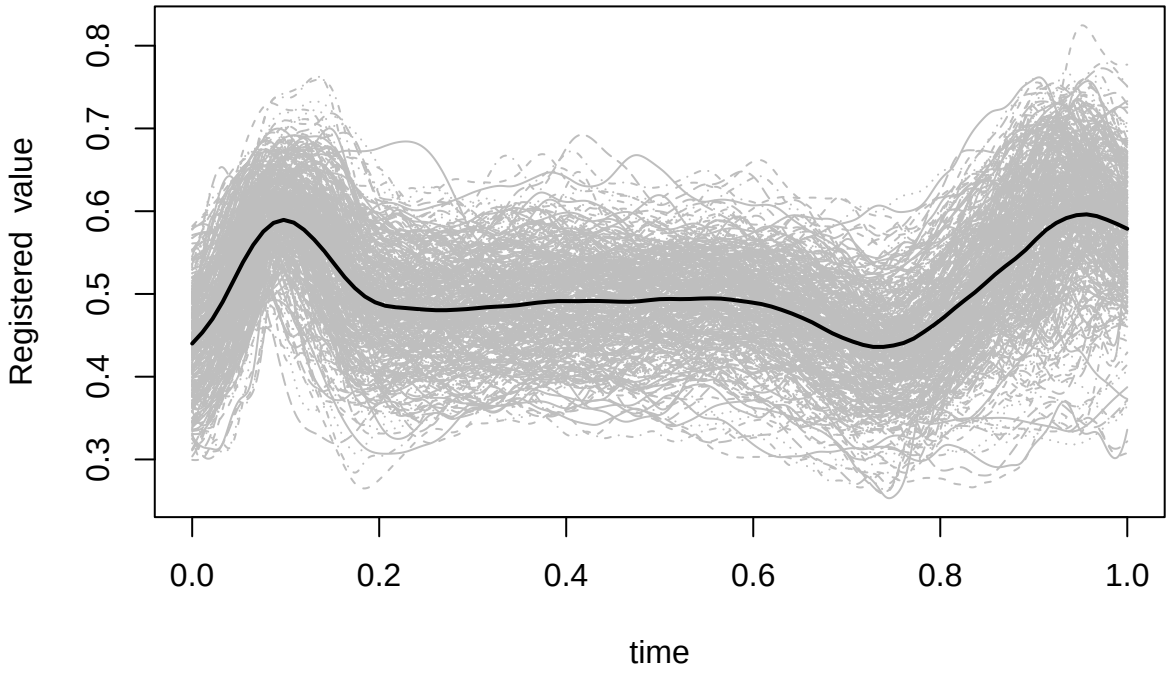
(b) Use *continuous registration* to align the curves. Plot the resulting curves and mean function. Comment on any differences from the plot in (a).

```
regList <- register.fd(data_smooth$fd)
```

```
conreg <- regList$regfd
plot(conreg,col='gray')
```

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

```
conreg_values <- eval.fd(argvals, data_smooth$fd)
lines(argvals,rowMeans(conreg_values),lwd=2) # mean function
```



(c) Carry out a PCA for both (a) and (b). Comment on any differences between FPCs and explained variance.

```
pca_a <- pca.fd(data_smooth$fd,nharm = 4)
pca_a$varprop
```

```
## [1] 0.63849341 0.08156312 0.06755725 0.06041105
```

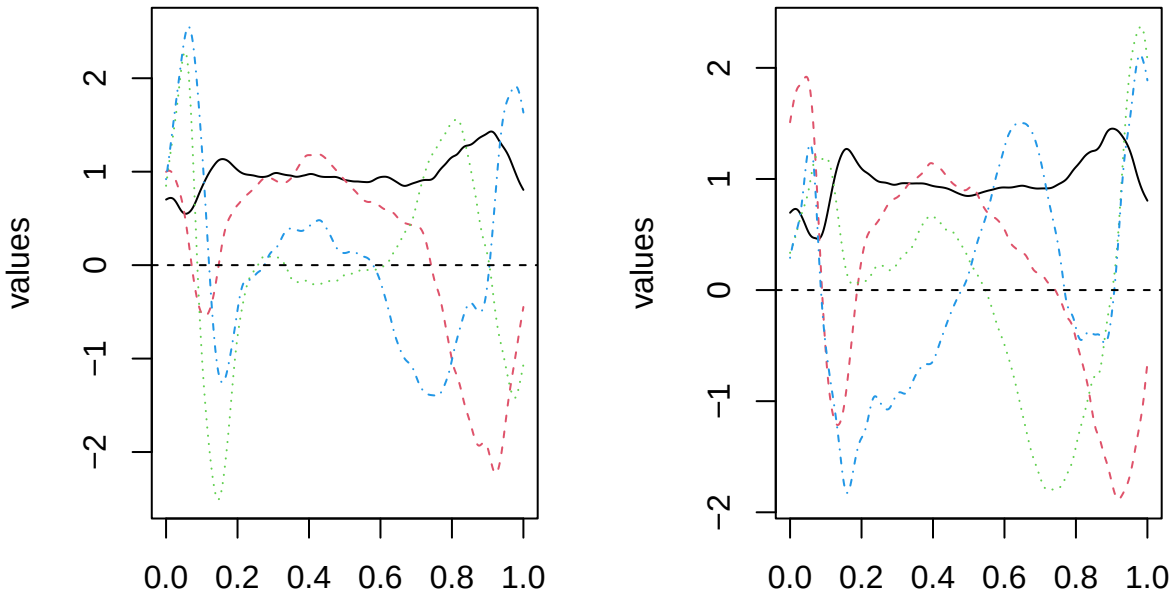
```
pca_b <- pca.fd(regList$regfd,nharm = 4)
pca_b$varprop
```

```
## [1] 0.69020417 0.07171080 0.05690093 0.04196608
```

```
par(mfrow=c(1,2))
plot(pca_a$harmonics)
```

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

```
plot(pca_b$harmonics)
```




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

(d) In your opinion, was curve alignment necessary for this data? Explain.

Not necessary

```
warpFunctions <- regList$warpfd
APList <- AmpPhaseDecomp(xfd=data_smooth$fd, # not aligned
                         yfd=conreg, # aligned
                         hfd=warpFunctions)

APList$RSQR
```

```
## [1] -0.06553882
```

```
APList$MS.amp
```

```
## [1] 0.004385622
```

```
APList$MS.pha
```

```
## [1] -0.0002697494
```

Paper : *Combining Registration and Fitting for Functional Model*

$$\begin{aligned} MS_{total} &= E\int [x(t)-\mu(t)]^2 dt \\ &= CE\int y^2(t)dt - \int \mu^2(t)dt \\ &= CE\int [y(t)-\nu(t)]^2 dt \\ &+ C\int \nu^2(t)dt - \int \mu^2(t)dt \end{aligned}$$

$$\begin{aligned} MS_{amp} &= CE\int [y(t)-\nu(t)]^2 dt \\ MS_{phase} &= C\int \nu^2(t)dt - \int \mu^2(t)dt \end{aligned}$$

If no registration, then C=1, $\mu = \nu$ and $MS_{phase} = 0$

Faulty registration may lead to $MS_{phase} \leq 0$ & $MS_{amp} > MS_{total} = 0$ $

2.5

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

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

```
rm(list=ls())
library(fda)
library(MASS)
```

(a) Simulate a functional sample over the unit interval each with a sample size of 50 from the Matern process. For the first half of the sample, set the mean function equal to 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 Matern 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.

```
# bump function
fx <- function(x,c0,r0,a0){
  if(abs(x-c0)<r0){
    a0*exp( -( 1 - ( ((x-c0)/r0))^2 )^(-1) )
  }else{0}
}
```

```
# Matern process sample
matern_cov <- function(t,s,sigma2,nu,rho){
  if(t==s){
    return(sigma2)
  } else{
    factor1 <- sigma2 / (gamma(nu) * 2^(nu-1))
    factor2 <- (sqrt(2*nu)*abs(t-s) / rho)^nu
    factor3 <- besselK(sqrt(2*nu)*abs(t-s)/rho, nu)
    return(factor1*factor2*factor3)
    }
} # each element in covariance

# Parameters for the Matérn process
sigma2 <- 1 # Variance
nu <- 1.5 # Smoothness parameter
rho <- 1 # Range parameter
time_points <- seq(0, 1, length.out = 50)  # 50 time points
n <- 50  # sample size

# create 50x50 matern_cov matrix
create_cov_mat <- function(time_points, sigma2, nu, rho){
  cov_mat <- matrix(0,n,n)
  for(i in 1:n){
    for(j in 1:n){
      cov_mat[i,j] <- matern_cov(time_points[i],
                                 time_points[j],
                                 sigma2, nu, rho)
    }
  }
  return(cov_mat)
```

```
}

cov_mat <- create_cov_mat(time_points,sigma2,nu,rho)
```

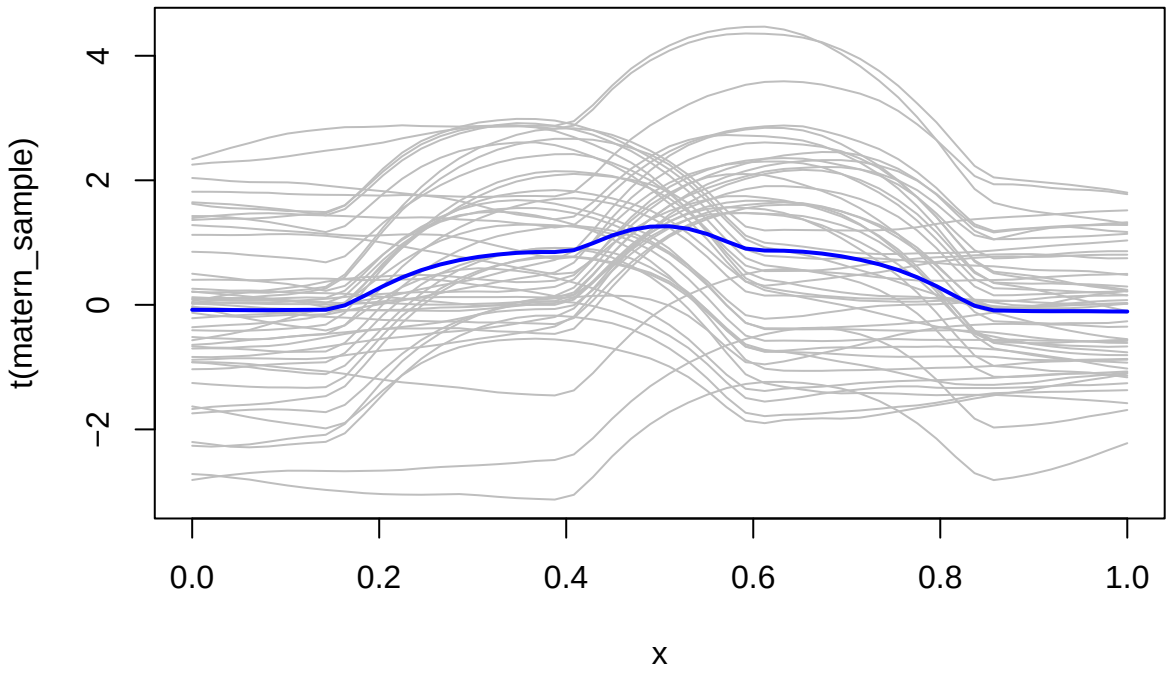
```
# bump function param
c0_1 <- 3/8
r0_1 <- 1/4
a0_1 <- 5
c0_2 <- 5/8
r0_2 <- 1/4
a0_2 <- 5

# bump function = mean function
bump_1 <- sapply(time_points, fx, c0 = c0_1, r0 = r0_1, a0 = a0_1)
bump_2 <- sapply(time_points, fx, c0 = c0_2, r0 = r0_2, a0 = a0_2)

matern_sample_1 <- mvrnorm(n/2,
                           mu=bump_1,
                           Sigma=cov_mat)
matern_sample_2 <- mvrnorm(n/2,
                           mu=bump_2,
                           Sigma=cov_mat)
dim(matern_sample_1)
```

```
## [1] 25 50
```

```
matern_sample <- rbind(matern_sample_1,matern_sample_2)
dim(matern_sample)
```

```
## [1] 50 50
```

```
# 50 time points, and each 50 matern samples

matplot(time_points,t(matern_sample),type='l',col='gray',lty=1)
lines(time_points, colMeans(matern_sample), col = 'blue', lwd = 2)
```

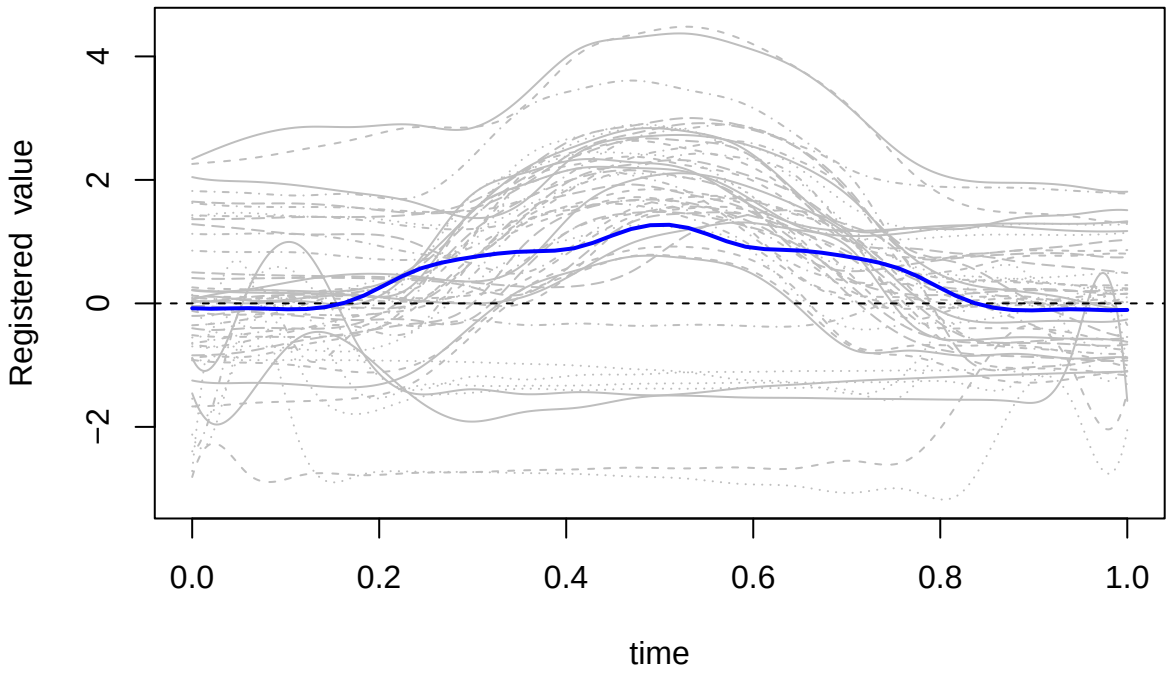


(b) Align the curves using continuous registration. Plot the resulting curves and include a mean function. Comment on any differences with (a) and if the registered curves exhibit any odd patterns.

```
basis <- create.bspline.basis(c(0,1), nbasis=20)
fd_obj <- Data2fd(time_points,t(matern_sample), basis)
```

```
regList <- register.fd(fd_obj)
conreg <- regList$regfd
```

```
plot(conreg,col='gray')

conreg_values <- eval.fd(time_points, fd_obj)
lines(time_points,rowMeans(conreg_values),lwd=2,col='blue')
```



(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.

```
pca_1 <- pca.fd(fd_obj,nharm=1)
pca_1$varprop
```

```
## [1] 0.7343449
```

```
pca_2 <- pca.fd(conreg,nharm=1)
pca_2$varprop
```

```
## [1] 0.8709556
```

```
# regression
x1 <- pca_1$scores # 50 x 4
y1 <- rep(c(0,1),each=25)
x2 <- pca_2$scores
y2 <- rep(c(0,1),each=25)

fit1 <- lm(x1~y1)
summary(fit1)
```

```
##
## Call:
## lm(formula = x1 ~ y1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.10615 -0.53721  0.06411  0.62289  2.34297
##
## Coefficients:
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.2874     0.2117  -1.358   0.1809
## y1            0.5749     0.2994   1.920   0.0608 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.059 on 48 degrees of freedom
## Multiple R-squared:  0.07133,    Adjusted R-squared:  0.05199
## F-statistic: 3.687 on 1 and 48 DF,  p-value: 0.06079
```

```
fit2 <- lm(x2~y2)
summary(fit2)
```

```
##
## Call:
## lm(formula = x2 ~ y2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3357 -0.4999  0.1392  0.5803  2.2745
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.2349     0.2345  -1.002    0.321
## y2            0.4699     0.3316   1.417    0.163
##
## Residual standard error: 1.172 on 48 degrees of freedom
## Multiple R-squared:  0.04015,    Adjusted R-squared:  0.02015
## F-statistic: 2.008 on 1 and 48 DF,  p-value: 0.1629
```

(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.