Department of Statistics and Data Science | Functional Data Analysis

Functional Linear Regression and Gaussian Process Regression

Gong Wenwu, 12031299

May 31, 2021
Emails: 12031299@mail.sustech.edu.cn

Abstract

In this report, there are two main points: *Functional Linear Regression* (fR) and *Gaussian Process Regression* (GPR). Firstly, I have performed the functional linear regression for the functional response *(Swedish)* w.r.t. scale and functional dependent variable. The purpose is to develop a predictive model for the Swedish log hazard rate and I have tried two regression methods. The results show that there is a late acceleration in the decline of the hazard rate and the concurrent model gives a good predictive power for Swedish data. Secondly, to deal with the functional data with multidimensional inputs, I have given a simulation study to discuss the Gaussian process regression and the results show that the trend and periodic pattern have been modeled.

Contents

1 Introduction

Functional regression depends on whether the responses or dependence are functional or vector data. There are many different aspects of functional regression and the main change is that regression coefficients now become regression coefficient functions with values $\beta(t)$. We should use others tools to model this kind of problems. In this report, the main idea is to build a useful predictive model for functional response data (**Swedish.data**: consist of the log hazard rates (instantaneous risk of death) at ages 0 to 80 for Swedish men by birth year from 1757 to 1900). From the functional linear modelling with regularized basis expansions fitted residual, I have founded that there is late acceleration in the decline of the log hazard rate. So, I have tried the concurrent model to get a more predictive performance. While parametric approaches are usually inflexible and restricted to special cases, nonparametric approaches commonly face the well-known curse of dimensionality. These difficulties can be aggravated when there is a multivariate response. To tackle these difficulties, functional data can be addressed by Gaussian process regression (GPR) models[(Shi)], where functional data are seen as realizations from a Gaussian process (GP) with a covariance kernel from a known parametric family. So, another point of this report is to discuss the Gaussian process regression and I have given a simulation study to illustrate the powerful of GPR.

2 Methodology

2.1 Functional-scale regression

For this kind of models, we want to answer the question that ”Does the shape of the mean hazard rate profile depend on birth year?” We may find out thees connections by using the coefficient functions given by functional-scale regression model. In this section, we will try to provide as many helpful tools (Similar with multivariate data and need only relatively obvious modifications to be adapted to the functional response context.) for analysis and inference as we can for this regression.

We have already noted, however, that the use of regularized basis function expansions gives us continuous control over smoothness while still permitting as much high frequency detail in the model as the data require [Ramsay2005]. The use of roughness penalties or regularization can play an important role in a functional linear model. In my report, I have modeled the functional-scale regression with regularized basis expansions and the relevant theory has been given as follows. Let us now assume that the observed functions y_i and regression functions $\beta(t)$ are expressed in basis expansion form, as the coefficients of a B-spline basis system (Note that we can use different nbasis, I smooth the hazard rate by using 38 B-spline basis with $\lambda = 0.01$ and expand the betabasis by using 32 B-spline basis). This means that

$$y(t) = C\phi(t), \tag{1}$$

where the N-vector y contains the N observed response functions, the K_y-vector ϕ contains the linearly independent basis functions, and the N by K_y matrix matrix C contains the coefficients of expansion of function y_i in its $i-th$ row. We can then obtain the following expressions for the penalized least squares criterion:

$$\mathrm{PENSSE}(y \mid \boldsymbol{\beta}) = \int (\mathrm{C}\phi - \mathbf{ZB}\boldsymbol{\theta})'(\mathrm{C}\phi - \mathbf{ZB}\boldsymbol{\theta}) + \lambda \int (L\mathbf{B}\boldsymbol{\theta})'(L\mathbf{B}\boldsymbol{\theta}). \tag{2}$$

We can use the **Kronecker product** to express the estimate $\hat{B}$ by setting the derivative of Eq. 2 with respect to matrix B and set the result to zero. Then the penalized least squares solution in this case is

$$\left[\left(\Theta'\Theta\right) \otimes \left(\mathbf{Z}'\mathbf{Z}\right) + \mathbf{R} \otimes \lambda\mathbf{I}\right] \ \mathrm{vec}\left(\mathbf{B}\right) = \left(\Theta' \otimes \mathbf{Z}'\right)\mathrm{vec}(\mathbf{Y}). \tag{3}$$

In my report, I have shown the process of tuning parameter λ , plotted the coefficient functions and assessed the model performance by using $R^2(t)$. So, I outline the relevant formula here to interpret what I have done. I select the amount of smoothing by leave-one-curve out cross validation, which defines as:

$$\mathrm{CV}(\lambda) = \sum \int \left(y_i(t) - \mathrm{z}_i\hat{\beta}_\lambda^{-i}(t)\right)^2 dt. \tag{4}$$

To assess the model and test significance, we can define $R^2(t)$ and pointwise F-statistic (creates a null distribution for a test of no effect) respectively.

$$R^2(t) = 1 - \frac{\sum\left(y_i(t) - \hat{y}_i(t)\right)^2}{\sum\left(y_i(t) - \bar{y}(t)\right)^2}, \quad F(t) = \frac{\mathrm{Var}(\hat{y}(t))}{\sum\left(y_i - \hat{y}_i\right)^2} \tag{5}$$

The confidence intervals of $\hat{\beta(t)}$ can be estimated by the standard error of a regression function at a value t. If the raw data are fit directly, the corresponding confidence interval is given by the residuals Σ_e, i.e.,

$$\mathrm{Var}[\mathrm{vec}(\hat{\boldsymbol{\beta}})] = \left[\Theta\left(\Theta'\Theta\right)^{-1}\Theta'\Sigma_e\Theta\left(\Theta'\Theta\right)^{-1}\Theta'\right] \otimes \left(\mathbf{Z}'\mathbf{Z}\right)^{-1}. \tag{6}$$

2.2 Functional-functional regression

Excepting for functional-scale regression, we also want to know How does a response variable profile depend on the associated covariate profile? In my report, the residual of functional-scale regression model fitted has shown a late acceleration in the decline of the hazard rate. So, we should use the concurrent model to make predictions. This model defines as:

$$\mathbf{y}(t) = \mathbf{Z}(t)\boldsymbol{\beta}(t) + \boldsymbol{\epsilon}(t), \tag{7}$$

where y is a functional vector of length N containing the response functions. Just same as Eq. 2, the weighted regularized fitting criterion is (p=1 for Swedish data)

$$\text{LMSSE}(\boldsymbol{\beta}) = \int \mathbf{r}(t)'\mathbf{r}(t)dt + \sum_{j}^{p} \lambda_j \int \left[L_j\beta_j(t)\right]^2 dt, \ \ \mathbf{r}(t) = \mathbf{y}(t) - \mathbf{Z}(t)\boldsymbol{\beta}(t). \tag{8}$$

We can write down the normal equations weighted least squares solution for the composite coefficient vector b:

$$\left[\int \Theta'(t)\mathbf{Z}'(t)\mathbf{Z}(t)\Theta(t)dt + \mathbf{R}\right]\mathbf{b} = \left[\int \Theta'(t)\mathbf{Z}'(t)\mathbf{y}(t)dt\right]. \tag{9}$$

2.3 Gaussian process regression

Let x be a functional variable and t be a Q -dimensional covariate. A nonparametric regression model is expressed as $x = f(\boldsymbol{t}) + \epsilon, \quad \epsilon \sim N\left(0, \sigma_\epsilon^2\right)$. However, most of the nonparametric methods suffer from the curse of dimensionality when they are applied to the problem with multi-dimensional covariates (i.e., Q is large). A variety of alternative approaches has been developed to overcome this problem. Examples include the additive model [Breiman1985], the varying-coefficient model [Hastie1993] and the GPR model [Shi2011]. The discrete form of a GPR model is defined as follows

$$x = f(\boldsymbol{t}) + \epsilon, \ \epsilon \sim N\left(0, \sigma_\epsilon^2\right)$$

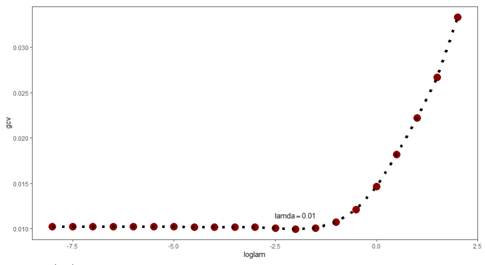
where $f(\cdot) \sim GP(\mu(\cdot), k(\cdot, \cdot))$ and $\text{COV}\left(f\left(t_i\right), f\left(\boldsymbol{t}_j\right)\right) = k\left(\boldsymbol{t}_i, \boldsymbol{t}_j\right)$ and $GP(\mu(\cdot), k(\cdot, \cdot))$ is a GP prior with mean function $\mu(\cdot)$ and covariance function $k(\cdot, \cdot)$. GP here can be treated as a prior of the unknown function $f(\cdot)$ from a Bayesian viewpoint. In my simulation study, I do some assumptions about model and then fit values and make predictions. We now temporarily assume that the noise variance σ_ϵ^2 is known, and the covariance function $k(\cdot, \cdot)$ is predetermined with fixed hyper-parameters known in advance. We use θ to denote σ_ϵ^2 and the hyper-parameters. They can be estimated by using, for example, the empirical Bayesian approach which will be discussed below. It is also common to assume a zero mean function, i.e., $\mu(\cdot) = 0$. In later section, I will give the **Powered exponential covariance functions** to use a GPR model to fit the simulated data and calculate the prediction.

3 Results

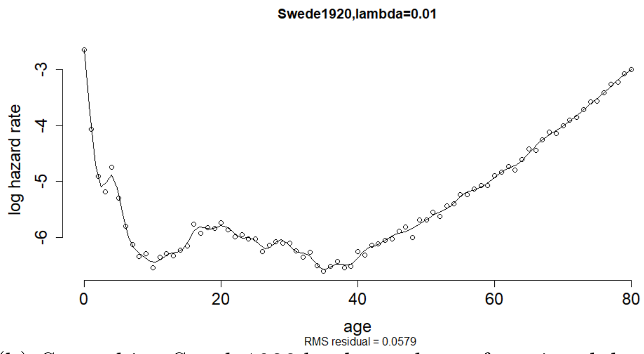
To get the following results, I have outlined the model building process: Firstly, I have **smoothed the functional log hazard rate** data by using roughness penalty and tuned it by using GCV (which can be guided by **Proj.A1**). Secondly, I have build the **regularized regression model** and choice λ_β by using cross-validation. Based on the fitted model, I have assessed the **model performance** by plotting the residual and calculating the **R square functions**. Also, to assess the influence of covariate, I have plotted the **coefficient functions along with confidence intervals**. Further, I have extrapolated the fitted models to **predict the hazard rate at 1920** and the concurrent model give a better predictive performance.

3.1 Smoothing functional data

lambda and nbasis We smoothed the Swedish data with a B-spline basis with 38 basis functions with roughness penalty $\lambda = 0.01$. The figures in Fig.1 are obtained using these functional responses. **The results show that the curves decrease over time and there also appears to be an increasing 'bump" in hazard rate around the late teenage years.**



(a) Chice of tuning parameter λ_y

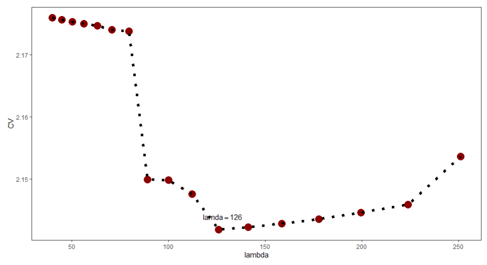


(b) Smoothing Swede1920 log hazard rate functional data

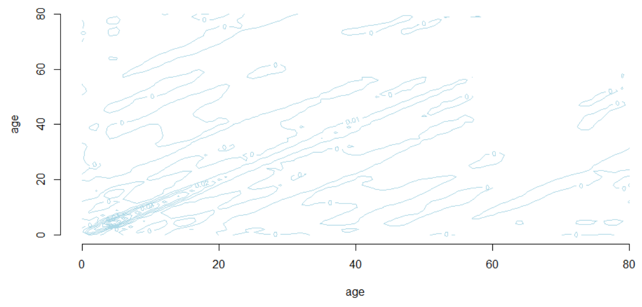
Fig. 1: Smoothing log hazard rate functional data.

3.2 Fitting the model

Guided by **Section 2**, we can use the R-package **fda** to build functional regression models. Firstly, I have built the **functional-scale regression model with nbetabasis (B-spline) equals to 42 and** $\lambda_\beta = 125.8925$. The standard errors can be calculated and the result shows there is **a downward trend of residuals.** To deal with problem, I have tried **add another covariate into model (such as the square of birth)**. However, there is some problems that I cannot fixed (Negative eigenvalue of coefficient matrix, maybe the betabasis is not right). And I have tried the concurrent model which can capture the late acceleration in the decline of the hazard and give a better predictive power. The follow figures have shown the mentioned results.

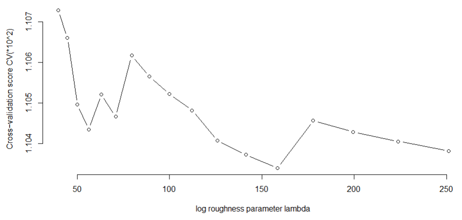


(a) Chice of tuning parameter λ_{beta}

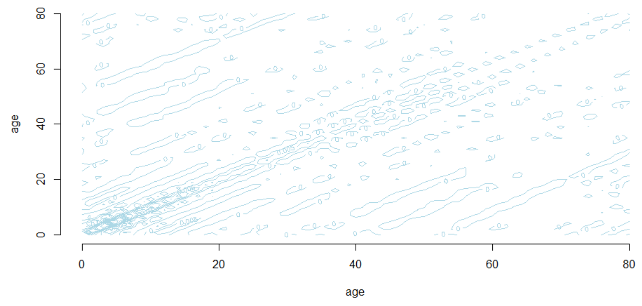


(b) Error covariance indicates lacking of fit

Fig. 2: Functional-scale regression model



(a) Chice of tuning parameter λ_{beta}

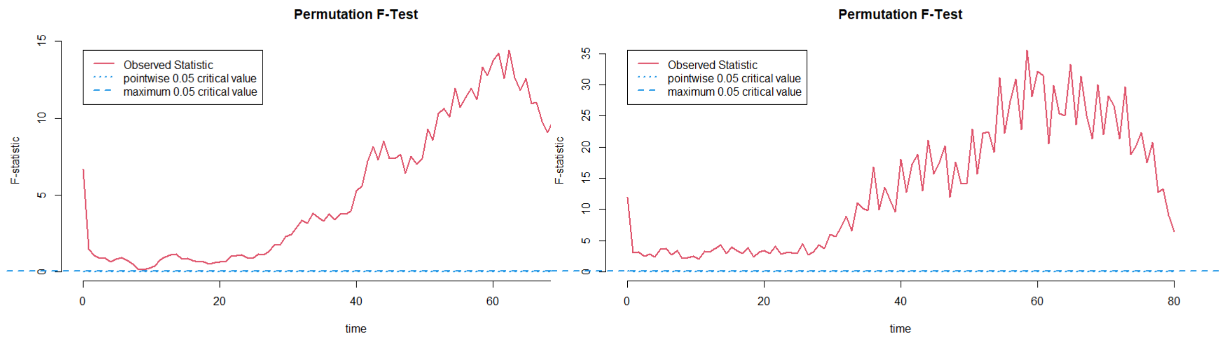


(b) Error covariance indicates fitting well

Fig. 3: Functional-functional (Concurrent) regression model

3.3 Assessing the fit

Based on the theory of **Section 2**, Fig. 4 have shown the **F-test value and the results have shown a clear predictability**(The horizontal dotted line indicates the 5% significance level for the F-distribution). Further, we have compared the $R^2(t)$ (see Fig. 5). The results shown that functional-scale model haven't captured the late acceleration in the decline of the hazard and **the concurrent model have a better predictive power(there is effect of time on hazard curves)**.



(a) Functional-scale model (b) Concurrent model

Fig. 4: Assessing the regression model by F-test

3.4 Confidence intervals for regression functions

In estimating and plotting the birth effects, we have taken our first step towards achieving the goal of **characterizing the typical hazard rate pattern for different birth year**. Fig. 6 have shown that the **curves decrease over time, there also appears to be an increasing 'bump" in mortality around the late teenage years and a 'backwards" movement of local peaks over the years.**

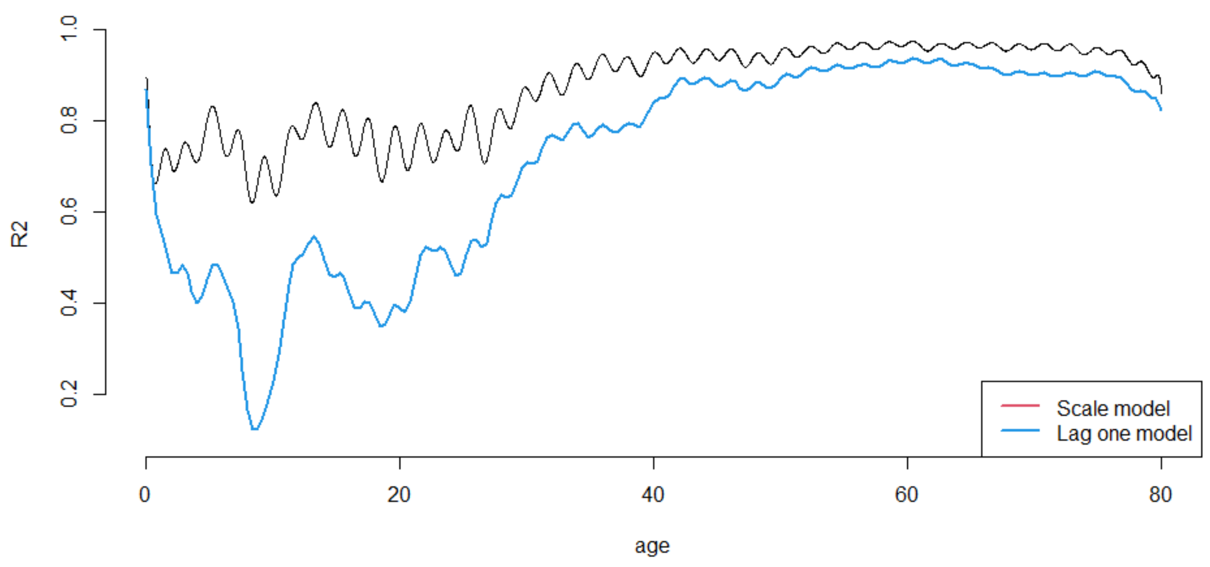
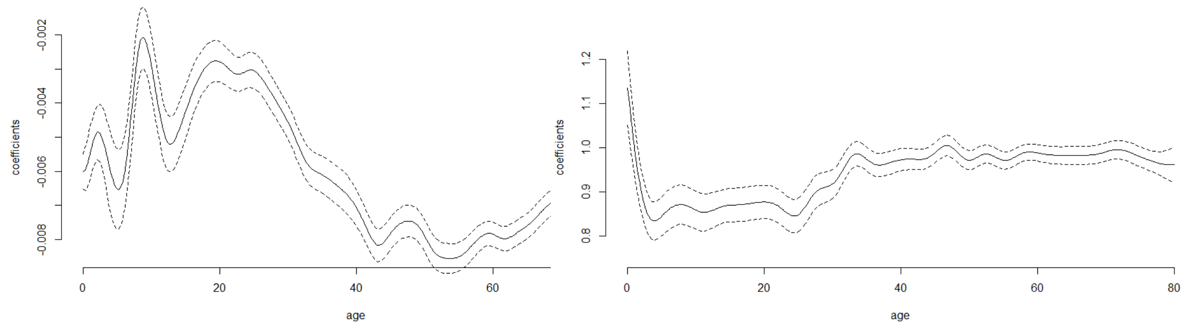


Fig. 5: Assessing the regression model by $R^2(t)$

3.5 Model Prediction

I have also extrapolate the models to predict the hazard rate at 1920 (Swede1920). Fig. 7 shown that concurrent model have a more powerful predictive and **give better predictions than just the mean hazard curve.**



(a) Functional-scale model (b) Concurrent model

Fig. 6: Confidence intervals for regression functions

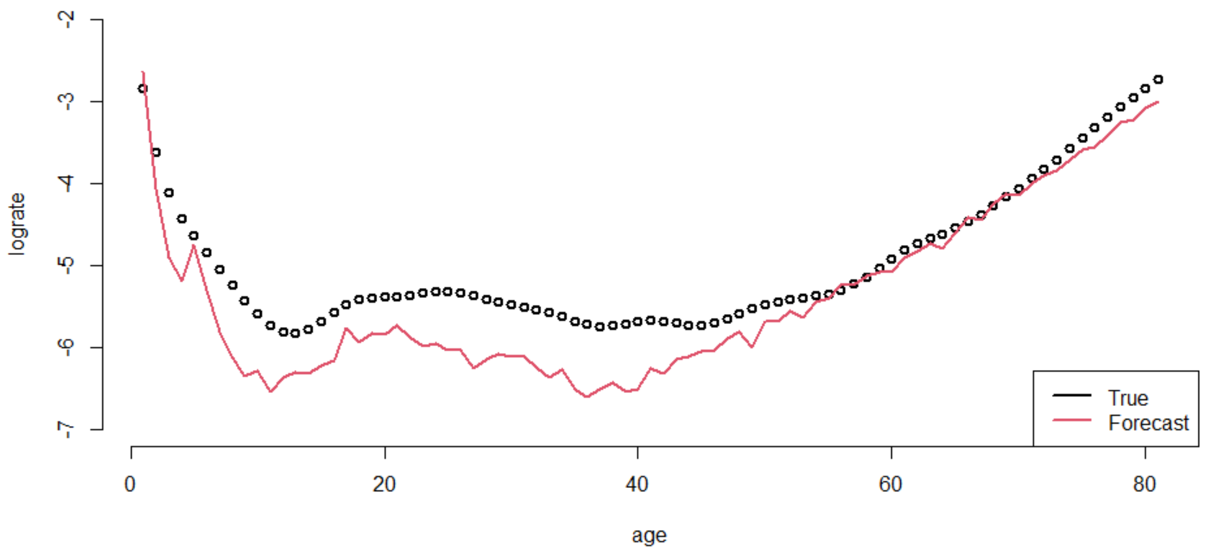
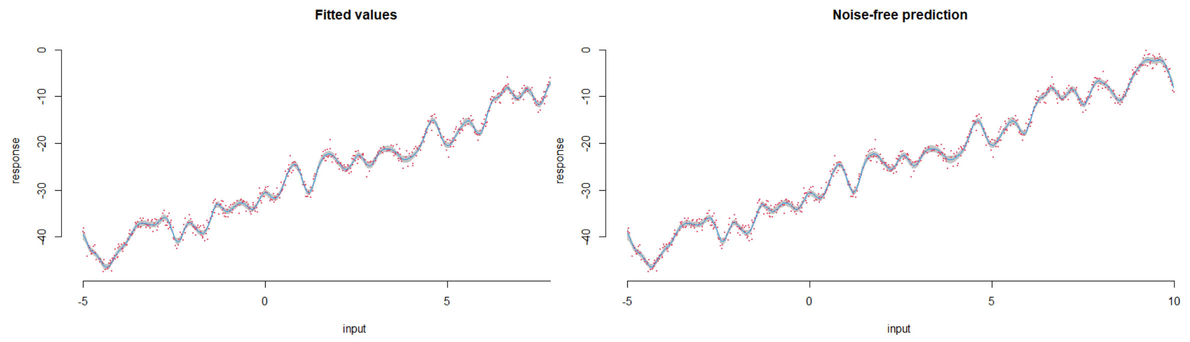


Fig. 7: The predictive power of concurrent model

3.6 A simulation study for GPR

I have generated 4-dimensional covariates data and split the data by 7:3. The simulation goal is to use a GPR model to fit the data and calculate the prediction. I have **used the squared exponential covariance function to model this trend and periodic pattern data.** Fig. 8 has shown the model performance and the ${R^2}_{os} = 0.99$ **shows the trained model has powerful predictive.**



(a) GPR model fitted (b) GPR model prediction

Fig. 8: A squared exponential covariance function GPR model

4 Conclusions

- The Swedish curves decrease over time and it appears to be an increasing 'bump" in log hazard rate around the late teenage years.
- There is a 'backwards" movement of local peaks over the years, i.e. a late acceleration in the decline of the hazard rate. And we should use the concurrent model to make predictions.
- The permuatation F-test and R^2 show that the concurrent model appears to be clear predictability.
- GPR model can fit the 4-dimensional covariate functional data (can fit the trend and periodic pattern successfully) and have a good predictive power.

References

[1] Ramsay, J.O. and Silverman, B.W.. Functional Data Analysis. (2005) Springer, New York.

[2] Wang, Jane-Ling and Chiou, Jeng-Min. Functional Data Analysis (June 2016). Annual Review of Statistics and Its Application, Vol. 3, Issue 1, pp. 257-295.

[3] Evandro Konzen and Yafeng Cheng and Jian Qing Shi. Gaussian Process for Functional Data Analysis: The GPFDA Package for R. arXiv.

[4] Breiman L and Friedman JH. Estimating Optimal Transformations for Multiple Regression and Correlation. Journal of the American Statistical Association, 80(391), 580–598.

[5] Hastie T and Tibshirani R Varying-Coefficient Models. Journal of the Royal Statistical Society B, 55(4), 757–779.

[6] Shi JQ and Choi T. Gaussian Process Regression Analysis for Functional Data. CRC.

Appendix

GWW 12031299

2021/5/27

Functional linear regression: functional-scale

```r
rm(list = ls())
library(fda)
library(ggplot2)
library(rgl)

setwd("E://R Files/Functional Data")
load('Data Sets/lifetable.Rdata')
birth = seq(1757,1900,length.out=144)
age = seq(0,80,length.out=81)
colnames(SwedeMat) = birth
rownames(SwedeMat) = age
Swede1900 = SwedeMat$"1900"
data = as.matrix(SwedeMat)
# plot(Swede1900) # no cycle
```

Smooth data using roughness penalty: B-spline

```r
bbreaks = seq(0,80,length.out=40) # tuning!
bbasis = create.bspline.basis(c(0,80), breaks=bbreaks, norder=4) # decide how much to smooth
gcv = c()
df = c()
sse = c()
Lfd = int2Lfd(2)
loglam = seq(-8,2,0.5)
nlam = length(loglam)
for (i in 1:nlam) {
  lambda = 10^loglam[i]
  D2fdPar = fdPar(bbasis,Lfdobj=Lfd,lambda=lambda)
  P_ageratefd = smooth.basis(age,data,D2fdPar)
  gcv[i] = mean(P_ageratefd$gcv) # we have many smoothers, average it!
  df[i] = mean(P_ageratefd$df)
  sse[i] = mean(P_ageratefd$SSE)
  mainstr = paste('lambda = ',lambda,' df = ',df[i],' gcv = ',gcv[i],sep='')
}
# plot(sse,type="b", xlab="log smoothing parameter lambda", ylab="sse")
res = data.frame(loglam,gcv,df,sse)
#ggplot(data=res,aes(loglam,gcv))+geom_point(color = "darkred",size=5)+geom_line(linetype="dotted",size
  #theme_bw()+theme(panel.grid=element_blank())+
  #annotate("text",x =-2, y = 0.0115,parse = T,label="lamda == 0.01")
```

```r
#ggplot(data=res,aes(loglam,sse))+geom_point(color = "darkred",size=5)+geom_line(linetype="dotted",size
  #theme_bw()+theme(panel.grid=element_blank())
which.min(gcv)
```

```
## [1] 13
```

```r
lambda_y = 10^loglam[which.min(gcv)]
D2fdPar = fdPar(bbasis,Lfdobj=Lfd,lambda=lambda_y) # important!
swedfd = smooth.basis(age,data,D2fdPar) # dependent variable
#plot(swedfd$fd,xlab="age", ylab="log hazard rate")
# plotfit.fd(Swededata,age,swedfd$fd)
par(mfrow=c(1,1), xpd=NA, bty="n")
P_ratefd = smooth.basis(age,Swede1920,D2fdPar) # smoother for Swede1920
#plotfit.fd(Swede1920,age,P_ratefd$fd,title="Swede1920,lambda=0.01",
          #xlab="age", ylab= "log hazard rate",cex.lab=1.5, cex.axis=1.5)
#plotfit.fd(Swede1920,age, P_ratefd$fd, residual = TRUE, titles = 'RMSE = 0.0579',
          #xlab="age", ylab= "log hazard rate", cex.lab=1.5, cex.axis=1.5)
P_RMSE = sqrt(mean((eval.fd(age, P_ratefd$fd) - Swede1920)^2))
print(P_RMSE)
```

```
## [1] 0.05789416
```

Functional regression: functional-scale

```r
## There is no regularized basis function expansion
ratefd = smooth.basis(age,data,D2fdPar)$fd # Step1: smooth functional dependent variable
p = 2
scaleList = vector("list", p) # initial the independent parameters (scale), important!
scaleList[[1]] = rep(1,144) # we have 144 objects
for (j in 2:p) {
  xj = birth # scale variables, numeric
  scaleList[[j]] = xj # the meaning of 1? Step 2
}
betabasis = create.bspline.basis(c(0,80), nbasis=28, norder=4) # how to determine the num of betabasis?
betafdPar = fdPar(betabasis) # add roughness penalty!
betaList = vector("list",p)
for (j in 1:p){
  betaList[[j]] = betafdPar # coefficient function! Step 3
}
fsRegressList = fRegress(ratefd, scaleList, betaList) # Smooth! Step 4
Rsq0 = 1 - mean.fd((ratefd-fsRegressList$yhatfdobj)^2)*mean.fd((center.fd(ratefd))^2)^(-1)
#plot(Rsq0, lwd=2,xlab="age",ylab="R2")
tfine = seq(0,80,by=0.5)
errvals = eval.fd(tfine,ratefd-fsRegressList$yhatfdobj)
# contour(tfine,tfine,errvals%*%t(errvals)/143,col='lightblue',xlab='age',ylab='age') # error covarianc
# for (j in 1:p){
#   plot(fsRegressList$betaestlist[[j]],col=j,main=paste("beta",j),xlab="age",ylab="coefficients")
# } # plot(fsRegressList$yhatfdobj$coefs,main="Prediction",xlab="age",ylab="log rate")

## Find out the lambda by CV (leave-one-out)
loglam = seq(1.6,2.4,0.05)
# zdata = cbind(scaleList[[1]],scaleList[[2]])
# CVdata = matrix(0,length(loglam),81)
# tbetalist = list()
# tZlist = list()
# for(i in 1:length(loglam)){
#   for(j in 1:81){
#     for(k in 1:2){
#       tbetalist[[k]] = fdPar(betabasis,2,10^loglam[i])
#       tZlist[[k]] = zdata[-j,k]
#     }
#     tres = fRegress(ratefd[-j],tZlist,tbetalist) # leave-one-out
#     yhat = 0
#     for(k in 1:2){
#       yhat = yhat + zdata[j,k]*tres$betaestlist[[k]]$fd # model fitted
#     }
#     err = ratefd[j] - yhat
#     CVdata[i,j] = inprod(err,err) # error covariance ?(n-p)
#   }
# }
# CV = apply(CVdata,1,mean)
# save(CV,file="E://R Files/Functional Data/Projects/A3/CVmat.RData")
lambda = 10^loglam
load('E://R Files/Functional Data/Projects/A3/CVmat.RData')
res = data.frame(lambda,CV)
# ggplot(data=res,aes(lambda,CV))+geom_point(color = "darkred",size=5)+geom_line(linetype="dotted",size
#   theme_bw()+theme(panel.grid=element_blank())+
#   annotate("text",x =128, y = 2.144,parse = T,label="lamda == 126")

yfdobj = ratefd
xnumobj = as.numeric(birth)
model1 = fRegress(yfdobj~xnumobj,data,method="model")
betabasis = create.bspline.basis(c(0,80), breaks=seq(0,80,length.out=40), norder=4)
betafd = fd(rep(1, 42), betabasis) # betalistlist
betafdPar = fdPar(betafd, Lfdobj=int2Lfd(2), lambda=lambda[which.min(CV)]) # roughness penalty given by
model1$betalist$xnumobj = betafdPar # replace the betalist, i.e. the regress coefficient function beta(
fslm_P1 = do.call('fRegress',model1)
# for (j in 1:2){
#   plot(fslm_P1$betaestlist[[j]],col=j,main=paste("beta",j),xlab="age",ylab="coefficients")
# }
Rsq_1 = 1 - mean.fd((ratefd-fslm_P1$yhatfdobj)^2)*mean.fd((center.fd(ratefd))^2)^(-1) # functional R2,
#plot(Rsq_1,lwd=2,xlab="age",ylab="R2") # intepret it!
errvals = eval.fd(age,ratefd-fslm_P1$yhatfdobj) # standard error
#persp3d(0:80,1:144,errvals,xlab='age',ylab='cohort',zlab='error',col='lightblue') # a downward trend o
Sigma = errvals%*%t(errvals)/143 # error covariance
#contour(age,age,Sigma,col='lightblue',xlab='age',ylab='age')
#Ftest = Fperm.fd(yfdobj,fslm_P1$xfdlist,fslm_P1$betalist) # creates a null distribution for a test of

## Model modification 1: wrong!!!
# xnum2obj = xnumobj^2 # xnumobj + 12
# model2 = fRegress(yfdobj~xnumobj+xnum2obj,data,method="model")
# betabasis2 = create.bspline.basis(c(0,80),nbasis=23, norder=4)
# betafd2 = fd(rep(1,23), betabasis2)
# betafdPar2 = fdPar(betafd2, Lfdobj=int2Lfd(2), lambda=lambda[which.min(CV)])
# model2$betalist$xnumobj = betafdPar2
# model2$betalist$xnum2obj = betafdPar2
# fslm_P2 = do.call('fRegress',model2) # Negative eigenvalue of coefficient matrix, Why???  may be beta
# # fslm_P2 = fRegress(yfdobj, c(fslm_P1$xfdlist,list((fslm_P1$xfdlist$xnumobj)^2)),c(fslm_P1$betalist,
# for (j in 1:3){
#   plot(fslm_P2$betaestlist[[j]],col=j,main=paste("beta",j),xlab="age",ylab="coefficients")
# }
# Rsq_2 = 1 - mean.fd((ratefd-fslm_P2$yhatfdobj)^2)*mean.fd((center.fd(ratefd))^2)^(-1)
# plot(Rsq_2,lwd=2,xlab="age",ylab="R2")
# errvals = eval.fd(age,ratefd-fslm_P2$yhatfdobj)
# persp3d(0:80,1:144,errvals,xlab='age',ylab='cohort',zlab='error',col='lightblue')
# Sigma = errvals%*%t(errvals)/142

## Prediction
yhatfd1920_s = fslm_P1$betaestlist[[1]]$fd + (1920-1757)*fslm_P1$betaestlist[[2]]$fd
yhatvals_1920_s = eval.fd(age,yhatfd1920_s)
#plot(Swede1920,ylim=c(-7,-2),col=1,lwd=2,xlab="age",ylab="lograte")
#plot(yhatvals_1920_s,col=2,lwd=2,xlab="age",ylab="lograte")

## Coefficient functions along with C.I.
y2cMap = smooth.basis(age,data,D2fdPar)$y2cMap # the value of y2cMap is given by smoothing dependent va
stderrList = fRegress.stderr(fslm_P1, y2cMap, Sigma) # formaula!
betastderrlist = stderrList$betastderrlist
# for (j in 1:p){
#   beta = fslm_P1$betaestlist[[j]]$fd
#   plot(beta, lwd=2,xlab="age",ylab="coefficients")
#   lines(beta + 2*betastderrlist[[j]],lty=2, lwd=1)
#   lines(beta - 2*betastderrlist[[j]], lty=2, lwd=1)
#   abline(h=0,lty=3)
# }# plotbeta(fslm_P$betaestlist,betastderrlist,age) # plot all beta coeffecients
```

Functional regression: functional-functional

```
ratefd = smooth.basis(age,data,D2fdPar)$fd
yfd = ratefd[2:144]
xfd = ratefd[1:143] # lag 1 concurrent?
model3 = fRegress(yfd~xfd,method="model")
betabasis = create.bspline.basis(c(0,80),norder=4,nbasis=32) # tuning it!
betafd = fd(rep(1, 32), betabasis)
# detach(package:fda)
# library(fda,lib.loc="D:/R/R-4.0.2/library/fdaold")
loglam = seq(1.6,2.4,0.05)
# nlam = length(loglam)
# SSE.CV = rep(0,nlam)
# for (i in 1:nlam){
#   lambda = 10^loglam[i]
#   betafdPar = fdPar(betafd, Lfdobj=int2Lfd(2),lambda=lambda)
#   model$betalist$xfd = betafdPar
#   flm_cv = do.call('fRegress.CV',model)
#   SSE.CV[i] = flm_cv$SSE.CV
# }
# min(SSE.CV)
# save(SSE.CV, file = "E://R Files/Functional Data/Projects/A3/fCVmat.RData") # there are many variable
# CV = ggplot(NULL,aes(loglam,SSE.CV/100))+geom_point(color = "darkred")+geom_abline(slope=1)+theme_bw(
# ggsave(CV,filename = "CV.png",width = 6,height = 4)
#detach(package:fda)
```

```r
#library(fda)
load('E://R Files/Functional Data/Projects/A3/fCVmat.RData')
#plot(10^loglam, SSE.CV/100, type="b", xlab="log roughness parameter lambda", ylab="Cross-validation sc
betafdPar = fdPar(betafd, 2,lambda=110.3395) # we only have one penalty, may be wrong!!!
model3$betalist$xfd = betafdPar
fflm_P = do.call('fRegress',model3)
## Assessing the fit
errmat = eval.fd(age, yfd - fflm_P$yhatfdobj)
Sigma = errmat%*%t(errmat)/144
# resid = data[,-1] - eval.fd(age,fflm_P$yhatfdobj)
# plot(resid[,143],cex.lab=1.5,cex.axis=1.5,xlab="age",ylab="residual")
# tfine = seq(0,80,by=0.5)
# yratemat = eval.fd(tfine,yfd)
# yratemeanvec = eval.fd(tfine,mean.fd(yfd))
# ratehatmat = eval.fd(tfine, fflm_P$yhatfdobj)
# resmat = yratemat - ratehatmat
# resmat0 = yratemat - yratemeanvec %*% matrix(1,1,143)
# SSE0 = apply((resmat0)^2, 1, sum)
# SSE1 = apply(resmat^2, 1, sum)
# Rsq_2 = (SSE0-SSE1)/SSE0
# plot(tfine[-1],Rsq_2[-1],xlab="age",ylab="R2")
# ## C.I.
# Sigma = var(t(resid))
#persp3d(age,1:143,errmat,xlab='age',ylab='cohort',zlab='error',col='lightblue') # better behaved than
Rsq_3 = 1 - mean.fd((yfd-fflm_P$yhatfdobj)^2)*mean.fd((center.fd(yfd))^2)^(-1)
#plot(Rsq_3,lwd=2,xlab="age",ylab="R2")
# contour(age,age,Sigma,col='lightblue',xlab='age',ylab='age')
# Ftest = Fperm.fd(yfd,fflm_P$xfdlist,fflm_P$betalist)
# plot(Rsq_3,ylim=c(0.1,1),col=3,lwd=2,xlab="age",ylab="R2")
# lines(Rsq_1,col=4,lwd=2)
# legend(x='bottomright',legend=c('Scale model','Lag one model'),lwd=2,col=c(2,4,3))
## C.I.
y2cMap = smooth.basis(age,data,D2fdPar)$y2cMap
stderrList = fRegress.stderr(fflm_P,y2cMap,Sigma)
betastderrlist = stderrList$betastderrlist
beta0 = fflm_P$betaestlist$const$fd
# plot(beta0,ylim=c(-0.95,0.5),xlab="age",ylab="coefficients", main="intercept")
# lines(beta0 + 2*betastderrlist[[1]],lty=2, lwd=1)
# lines(beta0 - 2*betastderrlist[[1]], lty=2, lwd=1)
# beta1 = fflm_P$betaestlist$xfd$fd
# plot(beta1,ylim=c(0.75,1.25),xlab="age",ylab="coefficients",main="One lag intercept")
# lines(beta1+2*betastderrlist[[2]],lty=2, lwd=1)
# lines(beta1-2*betastderrlist[[2]] ,lty=2, lwd=1)
```

```r
## Prediction for functional-functional
ratefd = smooth.basis(age,data,D2fdPar)$fd
yfd = ratefd[2:144]
xfd = ratefd[1:143]
betabasis_ = create.bspline.basis(c(0,80),norder=4,nbasis=23)
inbasis = eval.penalty(betabasis_,0) # part of penalty
inLbasis = eval.penalty(betabasis_,2)
R0 = inLbasis
Rs = inLbasis%x%inbasis
Rt = inbasis%x%inLbasis
```

```r
xphi = inprod(betabasis_,xfd) # integral
sxphi = apply(xphi,1,sum) # summation of objects, dimension is same with betabasis_
ypsi = inprod(betabasis_,yfd)
sypsi = apply(ypsi,1,sum)
yxphi = (xphi%x%matrix(1,23,1))*(matrix(1,23,1)%x%ypsi) # beta*integral
yxphi = apply(yxphi,1,sum)
inbetay = inprod(betabasis_,bbasis) # betabasis_*smooth funtional data
X = xphi%*%t(xphi)
Xmat = rbind( cbind( inbasis, t(sxphi)%x%inbasis ),
              cbind( sxphi%x%inbasis, X%x%inbasis) ) # important
lambda0 = 1e-5
lambdas = 1e0
lambdat = 1e0
penmat = rbind( cbind(lambda0*R0, matrix(0,23,23^2)),
                cbind(matrix(0,23^2,23),lambdas*Rs+lambdat*Rt)) # important
ymat = c(sypsi,yxphi)
betacoefs = solve(Xmat+penmat,ymat) # beta1coefs = fflm_P$betaestlist[[2]]$fd$coefs connections?
beta1Cmat = matrix(betacoefs[23+(1:23^2)],23,23) # important beta(s,t)

xfd_1900 = swedfd$fd[2:144]
xphi_1901 = inprod(betabasis_,xfd_1900)[,143]
yhatfd_1901_f = fd(betacoefs[1:23]%x%matrix(1,1,1),betabasis_) +fd(beta1Cmat%*%xphi_1901,betabasis_)
pre1901_f = eval.fd(age,yhatfd_1901_f)

yhatfd_last = yhatfd_1901_f
for (i in 1:20){
  xfd_new = yhatfd_last
  xphi_new=inprod(betabasis_,xfd_new)
  yhatfd_new = fd(betacoefs[1:23]%x%matrix(1,1,1),betabasis_) +fd(beta1Cmat%*%xphi_new,betabasis_)
  yhatfd_last=yhatfd_new
}
yhatvals_1920_f=eval.fd(age,yhatfd_new)
# plot(yhatvals_1920_f,ylim=c(-7,-2),xlab="age",ylab="lograte",col=1,lwd=2)
# lines(Swede1920,col=2,lwd=2)
# legend(x='bottomright',legend=c('True','Forecast'),lwd=2,col=c(1,2))
```

Gaussian process regression

```r
library(GPFDA)
library(MASS)

set.seed(12345)
X=matrix(1:4000,1000,4)
X[,1]=seq(-5,10,len=1000);X[,2]=seq(0,1,len=1000);X[,3]=seq(-15,-10,len=1000);X[,4]=seq(1,2,len=1000)
y = 0.2*X[,1] * abs(X[,1])^(1/3) - 4*sin(X[,2]) + exp(X[,3]) + log(X[,4])
hp = list('linear.a'=rep(log(10),4),'linear.i'=log(10),'pow.ex.w'=rep(log(10),4),'pow.ex.v'=log(5),'vv'=
Sigma = cov.linear(hyper=hp, input=X)+cov.pow.ex(hyper=hp,input=X,gamma=2)+diag(exp(hp$vv),1000)
epsilon = mvrnorm(n=1, mu=rep(0,1000), Sigma=Sigma)
y = y+epsilon


idx = 1:1000
ntest = 350
```

```
idx1 = sort(sample(1:1000, 350))
whichTest = idx%in%idx1
Xtest = X[whichTest, ]
ytest = y[whichTest]
Xtrain = X[!whichTest, ]
ytrain = y[!whichTest]

# kernel = c("linear","pow.ex")
# powfit = gpr(input=Xtrain,response=ytrain,kernel,trace=2)
# save(powfit,file='E://R Files/Functional Data/Projects/A3/powfit.RData')
load('E://R Files/Functional Data/Projects/A3/powfit.RData')
unlist(sapply(powfit$hyper,exp))
```

```
##     linear.a1     linear.a2     linear.a3     linear.a4      linear.i
##  3.716285e+00  1.941372e-02  5.260774e+00 1.957120e-105  1.605288e-03
##      pow.ex.v     pow.ex.w1     pow.ex.w2     pow.ex.w3     pow.ex.w4
##  6.506983e+00  1.055656e+01  2.029424e-01  6.217141e-01  1.694929e+01
##            vv
##  9.393879e-01
```

```
#plot(powfit)

predGPR = gprPredict(train=powfit,inputNew=Xtest, noiseFreePred=T)
#plot(predGPR)

# plot(ytest, predGPR$pred.mean, xlim=range(c(ytest, predGPR$pred.mean)), ylim=range(c(ytest, predGPR$p
# lines(x=range(c(ytest, predGPR$pred.mean)), y=range(c(ytest, predGPR$pred.mean)), lwd=2, col=4)
R2_gpr = 1-sum((predGPR$pred.mean - ytest)^2)/sum((mean(ytest) - ytest)^2)
```