Penalized Functional Binary Regression: An Application to PBC Dataset

Minjie Fan
Instructor: Hans-Georg Müller

March 19, 2013

1 Introduction

Functional binary regression models have proven useful for longitudinal data, in which predictors are usually sparse and irregular. Müller (2005) applied the extension of functional binary regression models to the data called primary biliary cirrhosis(PBC). The data comes from a Mayo Clinic trial conducted from 1974 to 1984. There are various sparsely and irregularly sampled covariates, and survival or censoring time for 312 patients in the data. The aim is to predict long-term survival based on a series of sparse initial measurements. In his paper, Müller merely considered serum bilirubin measurements as predictor, based on which he got the misclassification rate 26.54%. Actually, the other predictors contained in the data are also closely related with the disease, such as albumin and prothrombin time. Therefore, it is necessary to revisit this data and consider all the predictors in the model, for the purpose of decreasing the misclassification error.
Variables in the dataset, see Appendix A.

2 Functional data analysis framework

2.1 Generalized functional linear model

Assume n i.i.d. observations are $(Z_i^1, \cdots, Z_i^p, \{X_i^1(t), t \in \mathcal{T}_1\}, \cdots, \{X_i^q(t), t \in \mathcal{T}_q\}, Y_i), i = 1, \cdots, n$, where Z_i^j's are random predictor variables, $X_i^j(t)$'s are random predictor curves and Y_i's are random dependent variables. $E(X_i^j(t)) = \mu^j(t)$. Suppose there is a link function $g(\cdot)$ and a variance function $\sigma^2(\cdot)$. Then a generalized functional linear model or functional quasi-likelihood model is defined as:

$$E(Y_i|Z_i^j, X_i^k(t), j = 1, \cdots, p, k = 1, \cdots, q) = \mu_i,$$

$$Var(Y_i|Z_i^j, X_i^k(t), j = 1, \cdots, p, k = 1, \cdots, q) = \sigma^2(\mu_i),$$

$$g(\mu_i) = \eta_i = \alpha + \beta_1 Z_i^1 + \cdots + \beta_p Z_i^p + \int_{\mathcal{T}_1} (X_i^1(t) - \mu^1(t))\beta_1(t)dt + \cdots + \int_{\mathcal{T}_q} (X_i^q(t) - \mu^q(t))\beta_q(t)dt,$$

$$Y_i = g^{-1}\left(\alpha + \beta_1 Z_i^1 + \cdots + \beta_p Z_i^p + \int_{\mathcal{T}_1} (X_i^1(t) - \mu^1(t))\beta_1(t)dt + \cdots + \int_{\mathcal{T}_q} (X_i^q(t) - \mu^q(t))\beta_q(t)dt\right) + \epsilon_i. \tag{1}$$

This model is an extension of the one proposed by Müller and Stadtmüller (2005), which merely contains one random predictor curve $X(t)$.

2.2 Functional principal component analysis

Assume the random curve $X_i^j(t) \in L^2(\mathcal{T}_j)$, abbreviated as $X(t)$ hereinafter. $\mu(t) = E(X(t))$, and $G(s,t) = Cov\{X(s), X(t)\}$. Define the auto-covariance operator

$$(A_G f)(t) = \int f(s)G(s,t)ds,$$

and assume its orthonormal eigenfunctions Φ_k and ordered eigenvalues λ_k exist. Then $G(s,t) = \sum_{k=1}^{\infty} \lambda_k \Phi_k(s)\Phi_k(t)$. By the K-L expansion,

$$X(t) = \mu(t) + \sum_{k=1}^{\infty} \xi_k \Phi_k(t), \tag{2}$$

where ξ_k's are uncorrelated random variables, called functional principal component scores, and

$$\xi_k = \int (X(t) - \mu(t))\Phi_k(t)dt.$$

Suppose $\beta_j(t)$ in (1) can be expanded as $\beta_j(t) = \sum_{k=1}^{\infty} \beta_k^j \Phi_k(t)$. Then, the linear predictor in (1) is simplified as

$$\eta_i = \alpha + \beta_1 Z_i^1 + \cdots + \beta_p Z_i^p + \sum_{k=1}^{\infty} \beta_k^1 \xi_k^1 + \cdots + \sum_{k=1}^{\infty} \beta_k^q \xi_k^q. \tag{3}$$

When dealing with sparse and irregular functional data, we use the nonparametric approach called Principal Analysis through Conditional Expectation (PACE), which is under the Gaussian assumptions, see Yao (2005) for details.

3 Fitting method

3.1 Selecting the number of functional principal components

For practical application, we truncate the infinite terms in (3) at $K_j, j = 1, \cdots, q$,

$$\eta_i = \alpha + \beta_1 Z_i^1 + \cdots + \beta_p Z_i^p + \sum_{k=1}^{K_1} \beta_k^1 \xi_k^1 + \cdots + \sum_{k=1}^{K_q} \beta_k^q \xi_k^q. \tag{4}$$

Define the fraction of variance explained (FVE) as $F(K) = \frac{\sum_{k=1}^{K} \lambda_k}{\sum_{k=1}^{\infty} \lambda_k}$. We select the smallest K such that $F(K)$ is larger than a pre-specified threshold, which is taken as 0.99.

3.2 Predictor selection by group lasso

The number of predictors, except the intercept, is $n = p + \sum_{i=1}^{q} K_i$. when n is large, these predictors compose a high-dimensional dataset. Therefore, it is necessary to do the predictor selection. Besides, the predictors from the same random predictor curve should be selected simultaneously.

Meier (2008) proposed the group lasso for logistic regression. Suppose the vector of parameters is $(\beta_0, \boldsymbol{\beta}^T)^T$, with dimension $p+1$, where β_0 is the intercept. $\boldsymbol{\beta}$ is grouped in G groups $\boldsymbol{\beta}_1, \cdots, \boldsymbol{\beta}_G$, with group sizes $\{\mathrm{df}_1, \cdots, \mathrm{df}_G\}$. The logistic group lasso estimator $\hat{\boldsymbol{\beta}}_\lambda$ is given by the minimizer of the convex function

$$S_\lambda(\boldsymbol{\beta}) = -l(\boldsymbol{\beta}) + \lambda \sum_{g=1}^{G} s(\mathrm{df}_g) ||\boldsymbol{\beta}_g||_2, \tag{5}$$

where $l(\cdot)$ is the log-likelihood function, df_g is the number of parameters in the g^{th} group, and the function $s(\mathrm{df})$ is used to rescale the penalty, which is taken as $\mathrm{df}_g^{1/2}$.

The value of λ can be chosen by k-fold Cross-validation, where k is taken as 10.

The design matrix $X = [X_1|\cdots|X_G]$ should be centralized and block-wise standardized before entering the model, where we assume X_g's are of full column rank.

4 Application to PBC dataset

4.1 Preprocessing

We include the patients who satisfy the following two conditions:

- Survived the first 910 days of the study;
- Survival status beyond 10 years was known.

Altogether, there are 260 patients satisfied these two conditions, of which 84 died between 910 and 3650 days, and 176 lived beyond 3650 days. Based on the several initial measurements during the first 910 days, we would like to predicting survival beyond 10 years after entering the study. Omitting the predictors who have missing values or are categorical, we select 7 predictors, which are drug, age, sex, serum bilirubin, albumin, prothrombin time (PT) and SGOT. Among them, drug, age and sex are random variables, while the remaining are random curves. The bilirubin and SGOT measurements are log-transformed. For the responses Y_i,

we use 0 to represent short-lived, and 1 to represent long-lived.
The outliers are removed w.r.t. the random curves one by one. We find that there are outliers in albumin and PT, see Appendix B, Figure 1 and 2. 4 short-lived and 3 long-lived patients are removed in total.

4.2 Model fitting

By PACE, we get the estimates of mean functions, covariance surface, and eigenfunctions for each of the four random predictor curves, see Appendix B, Figure 3-7. We find that the log(bilirubin), PT and log(SGOT) are much higher for the short-lived patients than for the long-lived ones; the albumin is much lower for the short-lived patients. Besides, the log(bilirubin), PT and log(SGOT) are increasing over time, and the albumin is decreasing over time, for short-lived patients. These results are consistent with the ones of clinic study. Therefore, the four random predictor curves are good bio-markers for PBC patients, which distinguish short-lived patients from long-lived ones.
By FVE criterion, serum bilirubin, albumin, PT and SGOT have 3,4,4 and 1 eigenfunctions after truncation, respectively. Combined with the 3 random predictor variables drug, age and sex, we get the design matrix X, which is a 253×15 matrix (doesn't include the intercept term).
We use the package grplasso in R software to realize the method of group lasso. The multiplicative grid search set of the penalty parameter λ is $\{\lambda_{\max}, 0.96\lambda_{\max}, \cdots, 0.96^{148}\lambda_{\max}, 0\}$. λ is selected by 10-fold Cross-validation.

4.3 Results

Table 1: Leave-one-out classification detailed results

	Long-lived (True)	Short-lived (True)
Long-lived (Classified)	156	40
Short-lived (Classified)	17	40

Using the leave-one-out prediction error criterion, the overall misclassification rate is 22.53%, which is smaller than the one in Müller (2005), 26.54%. By including more predictors and using the shrinkage method, we decrease the misclassification error efficiently. Table 1 shows the detailed results. The misclassification rate for the short-lived patients are 50%, while the one for the long-lived patients are 9.83%. This can be explained by the unbalanced sample sizes of the short-lived and long-lived categories.

The histogram of the optimal penalty parameter λ selected by Cross-validation in the 253-times leave-one-out fittings is shown in Appendix B, Figure 8. The plot is right-skewed

Table 2: Parameter estimates by group lasso

Predictors	Coefficients
Intercept	4.5510001249
1^{st} score of Bili	-0.0207121621
2^{nd} score of Bili	-0.0125968522
3^{rd} score of Bili	0.0084731104
1^{st} score of Alb	0.0674235645
2^{nd} score of Alb	-0.1327425617
3^{rd} score of Alb	0.1218763442
4^{th} score of Alb	-1.7952969754
1^{st} score of PT	0.0004362802
2^{nd} score of PT	0.0550491950
3^{rd} score of PT	0.2886511832
4^{th} score of PT	-0.0769774141
1^{st} score of SGOT	-0.0325174314
drug	-0.1434167985
age	-0.0708683351
sex	0.0742933828

Table 3: The estimated importance degrees of all the predictors in decreasing order

serum bilirubin
age
prothrombin time
SGOT
albumin
sex
drug

and most λ's are between 0 and 0.25. Fitting the model by all the observations, we get the optimal λ, which is 0.1681001, by Cross-validation. The fitted coefficients are listed in table 2. The estimated standard errors of these fitted coefficients can be obtained by bootstrapping. The interpretation of the signs of these fitted coefficients is not reliable because the primary aim of the group lasso is to select the optimal set of predictors, not to find the predictors that give best explanation. However, taking a look at the access order of all the predictors into the model with λ varying from $\lambda_{\max}$ to 0, we can rate their importance degrees according to the order. Table 3 shows the estimated importance degrees of all the predictors in decreasing order. This makes sense because serum bilirubin concentration is one of the most important indicators of chronic liver cirrhosis, such as PBC, and the survival time is closely related with the age of the patient.

4.4 Discussion

Functional data analysis provides an innovative approach to handle random trajectories and infinite-dimensional data. Like, the classification of longitudinal data can be realized by functional binary regression, even in the case of sparse and irregular predictors. However, the problem of predictor selection still exists, especially for the data that have thousands or millions of random predictor variables and random predictor curves, such as fMRI images and gene sequences. Combined shrinkage methods with functional data analysis is a possible solution to this problem.

References

[1] Meier, L., Van De Geer, S. and Bühlmann, P. 2008. The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B*, **70**, 53–71.

[2] Müller, H.-G. and Stadtmüller, U. 2005. Generalized functional linear models. *Annals of statistics*, **33**, 774–805.

[3] Müller, H.-G. 2005. Functional modelling and classification of longitudinal data. *Scandinavian Journal of Statistics*, **32**, 223–240.

[4] Yao, F., Müller, H.-G. and Wang, J. 2005. Functional data analysis for sparse longitudinal data. *Journal of the American Statistical Association*, **100**, 577–590.

5 Appendix

5.1 Appendix A

Variables:
case number
number of days between registration and the earlier of death, transplantation, or study analysis time
status: 0=alive, 1=transplanted, 2=dead
drug: 1=D-penicillamine, 0=placebo
age in days, at registration
sex: 0=male, 1=female
day: number of days between enrollment and this visit date, remaining values on the line of data refer to this visit.
presence of asictes: 0=no 1=yes
presence of hepatomegaly: 0=no 1=yes
presence of spiders: 0=no 1=yes
presence of edema: 0=no edema and no diuretic therapy for edema; .5=edema present without diuretics, or edema resolved by diuretics; 1=edema despite diuretic therapy
serum bilirubin in mg/dl
serum cholesterol in mg/dl
albumin in gm/dl
alkaline phosphatase in U/liter
SGOT in U/ml (serum glutamic-oxaloacetic transaminase, the enzyme name has subsequently changed to ALT in the medical literature)
platelets per cubic in ml/1000
prothrombin time in seconds
histologic stage of disease

5.2 Appendix B

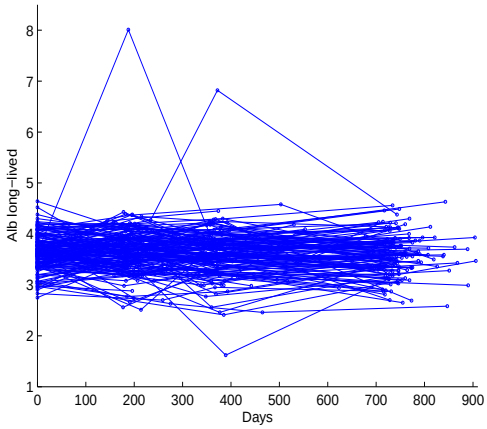

Figure 1: The observed initial albumin measurements, for 176 long-lived patients.

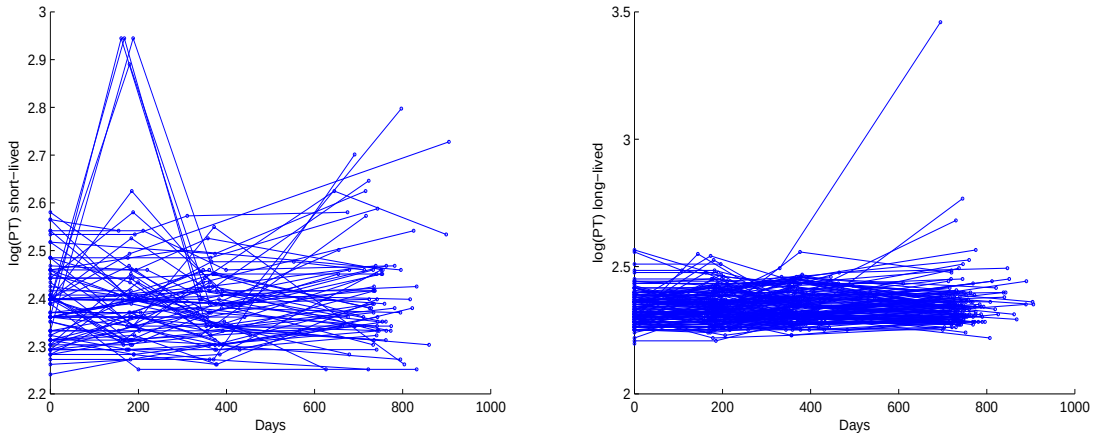

Figure 2: The observed initial log(PT) measurements (Left: for 84 short-lived patients, Right: for 174 long-lived patients).

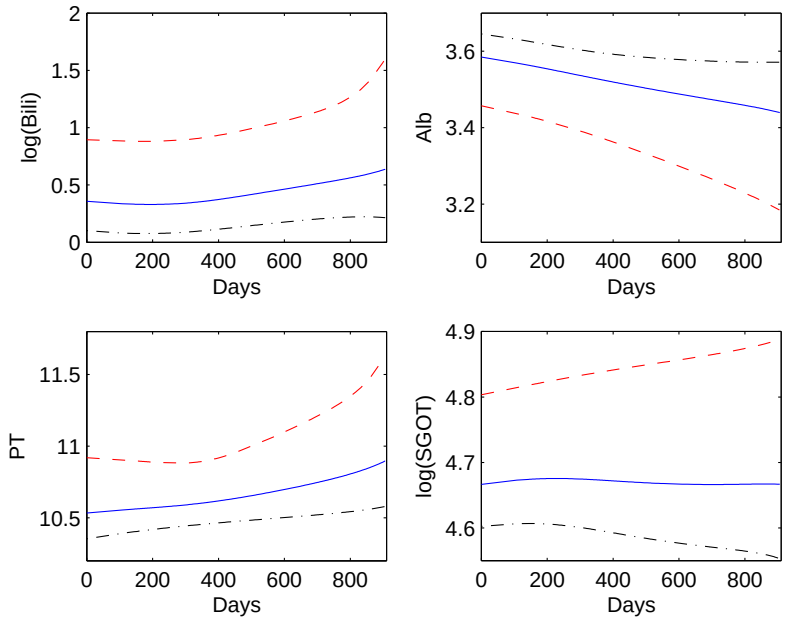

Figure 3: Smoothed mean functions for four random predictor curves (solid: all the 253 patients, dashed: the short-lived patients, dotdashed: the long-lived patients).

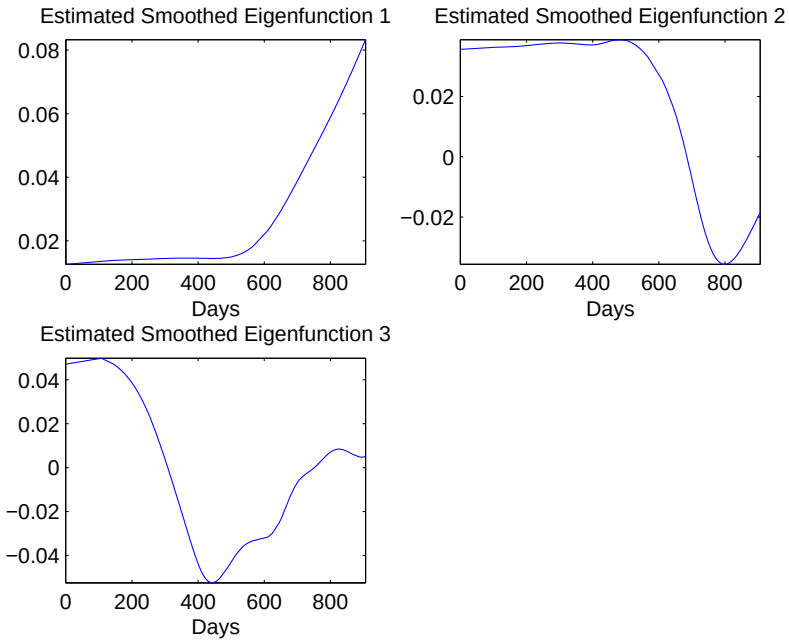

Figure 4: Smoothed eigenfunctions for serum bilirubin.

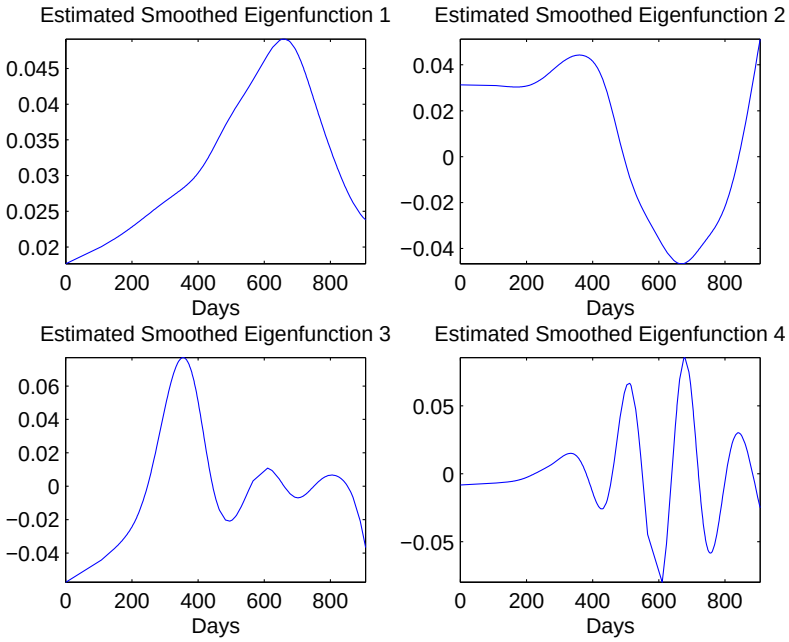

Figure 5: Smoothed eigenfunctions for albumin.

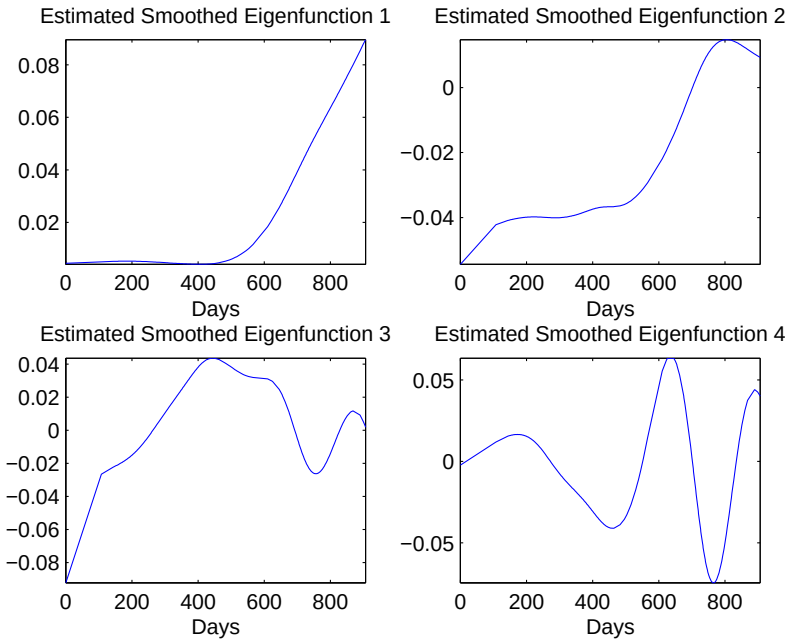

Figure 6: Smoothed eigenfunctions for PT.

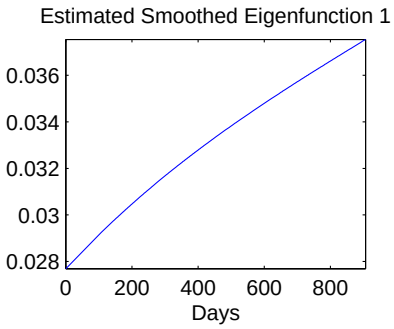

Figure 7: Smoothed eigenfunctions for SGOT.

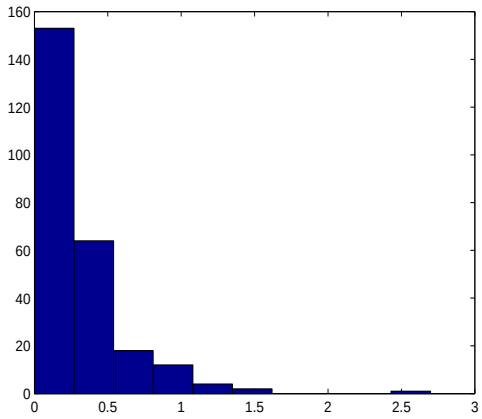

Figure 8: The histogram of the optimal penalty parameter λ selected by Cross-validation in the 253-times leave-one-out fittings.

5.3 Appendix C

Main.m (Matlab)

```
clear
clc
load('PBC.mat')

%pre-processing
PBC_910=PBC((PBC(:,2)>=910),:);
PBC_died=PBC_910((PBC_910(:,3)==2).*(PBC_910(:,2)<=3650)==1,:);
index_died=unique(PBC_died(:,1));
PBC_live=PBC_910((PBC_910(:,3)~=2)+(PBC_910(:,2)>3650)>=1,:);
index_live=unique(PBC_live(:,1));

PBC_died_910=PBC_died(PBC_died(:,7)<=910,:);
PBC_live_910=PBC_live(PBC_live(:,7)<=910,:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%plot log(bili) w.r.t short-lived
subplot(2,1,1)
hold on
for i=1:length(index_died)
    index_sub=index_died(i);
    PBC_sub=PBC_died_910(PBC_died_910(:,1)==index_sub,:);
    plot(PBC_sub(:,7),log(PBC_sub(:,12)),'-o','MarkerSize',2)
end
axis([0 910 -3 4])
xlabel('Days')
ylabel('log(Bili) short-lived')

%plot log(bili) w.r.t long-lived
subplot(2,1,2)
hold on
for i=1:length(index_live)
    index_sub=index_live(i);
    PBC_sub=PBC_live_910(PBC_live_910(:,1)==index_sub,:);
    plot(PBC_sub(:,7),log(PBC_sub(:,12)),'-o','MarkerSize',2)
end
axis([0 910 -3 4])
xlabel('Days')
ylabel('log(Bili) long-lived')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```
%plot alb w.r.t short-lived
subplot(2,1,1)
hold on
for i=1:length(index_died)
    index_sub=index_died(i);
    PBC_sub=PBC_died_910(PBC_died_910(:,1)==index_sub,:);
    plot(PBC_sub(:,7),PBC_sub(:,14),'-o','MarkerSize',2)
end
axis([0 910 1.5 5])
xlabel('Days')
ylabel('Alb short-lived')

%remove the outlier
index_live=remove_outlier(PBC_live_910,6.5,index_live,14,false);

%plot alb w.r.t long-lived
subplot(2,1,2)
hold on
for i=1:length(index_live)
    index_sub=index_live(i);
    PBC_sub=PBC_live_910(PBC_live_910(:,1)==index_sub,:);
    plot(PBC_sub(:,7),PBC_sub(:,14),'-o','MarkerSize',2)
end
axis([0 910 1.5 5])
xlabel('Days')
ylabel('Alb long-lived')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%remove the outlier
index_died=remove_outlier(PBC_died_910,2.85,index_died,18,true);

%plot PT w.r.t short-lived
subplot(2,1,1)
hold on
for i=1:length(index_died)
    index_sub=index_died(i);
    PBC_sub=PBC_died_910(PBC_died_910(:,1)==index_sub,:);
    plot(PBC_sub(:,7),PBC_sub(:,18),'-o','MarkerSize',2)
end
axis([0 910 8 18])
xlabel('Days')
ylabel('PT short-lived')

%remove the outlier
index_live=remove_outlier(PBC_live_910,3,index_live,18,true);

%plot PT w.r.t long-lived
subplot(2,1,2)
hold on
for i=1:length(index_live)
    index_sub=index_live(i);
    PBC_sub=PBC_live_910(PBC_live_910(:,1)==index_sub,:);
    plot(PBC_sub(:,7),PBC_sub(:,18),'-o','MarkerSize',2)
end
axis([0 910 8 18])
xlabel('Days')
ylabel('PT long-lived')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%plot log(SGOT) w.r.t short-lived
subplot(2,1,1)
hold on
for i=1:length(index_died)
    index_sub=index_died(i);
    PBC_sub=PBC_died_910(PBC_died_910(:,1)==index_sub,:);
    plot(PBC_sub(:,7),log(PBC_sub(:,16)),'-o','MarkerSize',2)
end
axis([0 910 3 7])
xlabel('Days')
ylabel('log(SGOT) short-lived')

%plot log(SGOT) w.r.t long-lived
subplot(2,1,2)
hold on
for i=1:length(index_live)
```

```
        index_sub=index_live(i);
        PBC_sub=PBC_live_910(PBC_live_910(:,1)==index_sub,:);
        plot(PBC_sub(:,7),log(PBC_sub(:,16)),'-o','MarkerSize',2)
    end
    axis([0 910 3 7])
    xlabel('Days')
    ylabel('log(SGOT) long-lived')

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %preliminary computation
    n_died=length(index_died);
    n_live=length(index_live);
    n_total=n_died+n_live;

    %set
    p=setOptions('selection_k','FVE','FVE_threshold',0.99);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %FPCA(Bili short-lived)
    [X_Bili_died,y_died,t_died]=my_FPCA(PBC_died_910,n_died,index_died,12,true);
    out1_Bili_died=getVal(X_Bili_died,'out1');
    mu_Bili_died=getVal(X_Bili_died,'mu');

    %FPCA(Bili long-lived)
    [X_Bili_live,y_live,t_live]=my_FPCA(PBC_live_910,n_live,index_live,12,true);
    out1_Bili_live=getVal(X_Bili_live,'out1');
    mu_Bili_live=getVal(X_Bili_live,'mu');

    %FPCA(Bili overall)
    y_tot=[y_died y_live];
    t_tot=[t_died t_live];
    X_Bili_tot=FPCA(y_tot,t_tot,p);  %output
    out1_Bili_tot=getVal(X_Bili_tot,'out1');
    mu_Bili_tot=getVal(X_Bili_tot,'mu');

    %plot
    subplot(2,2,1)
    plot(out1_Bili_died,mu_Bili_died,'--r')
    hold on
    plot(out1_Bili_live,mu_Bili_live,'-.k')
    plot(out1_Bili_tot,mu_Bili_tot,'b')
    axis([0 910 0 2])
    xlabel('Days')
    ylabel('log(Bili)')

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %FPCA(Alb short-lived)
    [X_Alb_died,y_died,t_died]=my_FPCA(PBC_died_910,n_died,index_died,14,false);
    out1_Alb_died=getVal(X_Alb_died,'out1');
    mu_Alb_died=getVal(X_Alb_died,'mu');

    %FPCA(Alb long-lived)
    [X_Alb_live,y_live,t_live]=my_FPCA(PBC_live_910,n_live,index_live,14,false);
    out1_Alb_live=getVal(X_Alb_live,'out1');
    mu_Alb_live=getVal(X_Alb_live,'mu');

    %FPCA(Alb overall)
    y_tot=[y_died y_live];
    t_tot=[t_died t_live];
    X_Alb_tot=FPCA(y_tot,t_tot,p);  %output
    out1_Alb_tot=getVal(X_Alb_tot,'out1');
    mu_Alb_tot=getVal(X_Alb_tot,'mu');

    %plot
    subplot(2,2,2)
    plot(out1_Alb_died,mu_Alb_died,'--r')
    hold on
    plot(out1_Alb_live,mu_Alb_live,'-.k')
    plot(out1_Alb_tot,mu_Alb_tot,'b')
    axis([0 910 3.1 3.7])
    xlabel('Days')
    ylabel('Alb')

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%FPCA(PT short-lived)
[X_PT_died,y_died,t_died]=my_FPCA(PBC_died_910,n_died,index_died,18,false);
out1_PT_died=getVal(X_PT_died,'out1');
mu_PT_died=getVal(X_PT_died,'mu');

%FPCA(PT long-lived)
[X_PT_live,y_live,t_live]=my_FPCA(PBC_live_910,n_live,index_live,18,false);
out1_PT_live=getVal(X_PT_live,'out1');
mu_PT_live=getVal(X_PT_live,'mu');

%FPCA(PT overall)
y_tot=[y_died y_live];
t_tot=[t_died t_live];
X_PT_tot=FPCA(y_tot,t_tot,p);  %output
out1_PT_tot=getVal(X_PT_tot,'out1');
mu_PT_tot=getVal(X_PT_tot,'mu');
getVal(X_PT_tot,'no_opt')

%plot
subplot(2,2,3)
plot(out1_PT_died,mu_PT_died,'--r')
hold on
plot(out1_PT_live,mu_PT_live,'-.k')
plot(out1_PT_tot,mu_PT_tot,'b')
axis([0 910 10.2 11.8])
xlabel('Days')
ylabel('PT')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%FPCA(SGOT short-lived)
[X_SGOT_died,y_died,t_died]=my_FPCA(PBC_died_910,n_died,index_died,16,true);
out1_SGOT_died=getVal(X_SGOT_died,'out1');
mu_SGOT_died=getVal(X_SGOT_died,'mu');

%FPCA(SGOT long-lived)
[X_SGOT_live,y_live,t_live]=my_FPCA(PBC_live_910,n_live,index_live,16,true);
out1_SGOT_live=getVal(X_SGOT_live,'out1');
mu_SGOT_live=getVal(X_SGOT_live,'mu');

%FPCA(SGOT overall)
y_tot=[y_died y_live];
t_tot=[t_died t_live];
X_SGOT_tot=FPCA(y_tot,t_tot,p);  %output
out1_SGOT_tot=getVal(X_SGOT_tot,'out1');
mu_SGOT_tot=getVal(X_SGOT_tot,'mu');
getVal(X_SGOT_tot,'no_opt')

%plot
subplot(2,2,4)
plot(out1_SGOT_died,mu_SGOT_died,'--r')
hold on
plot(out1_SGOT_live,mu_SGOT_live,'-.k')
plot(out1_SGOT_tot,mu_SGOT_tot,'b')
axis([0 910 4.55 4.9])
xlabel('Days')
ylabel('log(SGOT)')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%plot smoothed eigenfunctions
plot_eigenfun(X_Bili_tot)
plot_eigenfun(X_Alb_tot)
plot_eigenfun(X_PT_tot)
plot_eigenfun(X_SGOT_tot)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%pre-processing
PC_scores_Bili=getVal(X_Bili_tot,'xi_est');
PC_scores_Alb=getVal(X_Alb_tot,'xi_est');
PC_scores_PT=getVal(X_PT_tot,'xi_est');
PC_scores_SGOT=getVal(X_SGOT_tot,'xi_est');

drug=zeros(n_total,1);
for i=1:n_died
```

```
    drug(i,1)=unique(PBC_died_910(find(PBC_died_910(:,1)==index_died(i)),4));
end
for i=1:n_live
    drug(n_died+i,1)=unique(PBC_live_910(find(PBC_live_910(:,1)==index_live(i)),4));
end

age=zeros(n_total,1);
for i=1:n_died
    age(i,1)=unique(PBC_died_910(find(PBC_died_910(:,1)==index_died(i)),5))/365;
end
for i=1:n_live
    age(n_died+i,1)=unique(PBC_live_910(find(PBC_live_910(:,1)==index_live(i)),5))/365;
end

sex=zeros(n_total,1);
for i=1:n_died
    sex(i,1)=unique(PBC_died_910(find(PBC_died_910(:,1)==index_died(i)),6));
end
for i=1:n_live
    sex(n_died+i,1)=unique(PBC_live_910(find(PBC_live_910(:,1)==index_live(i)),6));
end

label=[zeros(n_died,1);ones(n_live,1)];
predictor=[PC_scores_Bili PC_scores_Alb PC_scores_PT PC_scores_SGOT drug age sex];
save('glm.mat','label','predictor','-v6')
```

my_FPCA.m (Matlab)

```
function [X,y,t]=my_FPCA(PBC_910,n,index,which,logtf)
y=cell(1,n);
if logtf==true
    for i=1:n
        y{i}=log(PBC_910(PBC_910(:,1)==index(i),which))';
    end
else
    for i=1:n
        y{i}=PBC_910(PBC_910(:,1)==index(i),which)';
    end
end
t=cell(1,n);
for i=1:n
    t{i}=PBC_910(PBC_910(:,1)==index(i),7)';
end
X=FPCA(y,t);
end
```

plot_eigenfun.m (Matlab)

```
function plot_eigenfun(X)
n=getVal(X,'no_opt');
phi=getVal(X,'phi');
phi_time=getVal(X,'out1');
for i=1:n
    subplot(2,2,i)
    plot(phi_time,phi(:,i))
    xlabel('Days')
    title(['Estimated Smoothed Eigenfunction ' num2str(i)])
    axis tight
end
end
```

remove_outlier.m (Matlab)

```
function [index,outlier]=remove_outlier(PBC_910,threshold,index,which,logtf)
if logtf==false
    outlier=unique(PBC_910(PBC_910(:,which)>threshold,1));
else
    outlier=unique(PBC_910(log(PBC_910(:,which))>threshold,1));
for i=1:length(outlier)
    index(find(index==outlier(i)))=[];
end
end
```

group_lasso.r (R)

```r
rm(list=ls())
library(R.matlab)
library(grplasso)
library(bootstrap)
dt<-readMat("D://glm.mat")
label<-dt$label
predictor<-dt$predictor
n<-nrow(predictor)
predictor<-cbind(matrix(rep(1,n),n,1),predictor)
index<-c(NA,1,1,1,2,2,2,2,3,3,3,3,4,5,6,7)

theta.predict<-function(fit.obj,x){
  predict(fit.obj,x,type="response")
}

cv.error<-function(tra.pre,tra.lab,lambda,k=10){
  index<-c(NA,1,1,1,2,2,2,2,3,3,3,3,4,5,6,7)
  list.val<-crossval(tra.pre,tra.lab,grplasso,theta.predict,index=index,
model=LogReg(),lambda=lambda,center=TRUE,standardize=TRUE,ngroup=k)
  mean(as.numeric(list.val$cv.fit>0.5)!=tra.lab)
}

lambda.max<-30
m<-148
search.set<-rep(0,m+2)
search.set[1]<-lambda.max
for (i in 2:(m+1)){
  search.set[i]<-search.set[i-1]*0.96
}
R<-10
pre.lab<-matrix(0,n,1)
record.min.lambda<-matrix(0,n,1)

for (i in 1:n){
  tra.pre<-predictor[-i,]
  tra.lab<-label[-i]
  min.error<-1
  for (j in 1:(m+2)){
    lambda<-search.set[j]
    s<-0
    for (r in 1:R){
      s<-s+cv.error(tra.pre,tra.lab,lambda)
    }
    s<-s/R
    if (s<min.error){
      min.error<-s
      min.lambda<-lambda
    }
  }
  record.min.lambda[i]<-min.lambda
  fit.obj<-grplasso(x=tra.pre,y=tra.lab,index=index,model=LogReg(),
lambda= min.lambda,center=TRUE,standardize=TRUE)
  pre.lab[i]<-as.numeric(predict(fit.obj,t(as.matrix(predictor[i,])),type="response")>0.5)
}
write.csv(pre.lab,"result1.csv")
write.csv(record.min.lambda,"result2.csv")

min.error<-1
for (j in 1:(m+2)){
  lambda<-search.set[j]
  s<-0
  for (r in 1:R){
    s<-s+cv.error(predictor,label,lambda)
  }
  s<-s/R
  if (s<min.error){
    min.error<-s
    min.lambda<-lambda
  }
}

fit.obj<-grplasso(x=predictor,y=label,index=index,model=LogReg(),
lambda=min.lambda,center=TRUE,standardize=TRUE)
fit.obj2<-glm.fit(x=predictor,y=label)
```