

Fitting functional Cox models in R using mgcv package

Erjia Cui

2/9/2021

Introduction

This document shows how to fit Additive Functional Cox Model (AFCM) (Cui, Crainiceanu, and Leroux 2020) and Linear Functional Cox Model (LFCM) (Gellar et al. 2015; Kong et al. 2018) using the mgcv package in R. The document has two parts. In the first part we simulate a dataset and fit both models step-by-step. In the second part we show model applications to a real NHANES dataset organized using the rnhanesdata (Leroux et al. 2019) package.

```
## load packages
library(mgcv)
library(refund)
```

Part 1: Simulated Dataset

We first simulate a dataset `data_analysis` which consists of:

1. `event`: a binary indicator of the event.
2. `survtime`: time to event (in year unit, for example).
3. `X`: a functional predictor stored in an $N \times S$ matrix. Here N is the number of subjects, S is the number of grid points on the functional domain. Each row contains the functional observations of one subject on S locations.
4. `Z`: other scalar predictors. We only include one scalar predictor in this example.

```
## simulate a dataset
set.seed(2021)
N <- 2000 ## number of subjects
S <- 1000 ## number of functional observations per subject
event <- rbinom(N, 1, 0.3) ## 30% of subjects have events observed
survtime <- runif(N, 0, 10) ## observed time
## transformations on X may be necessary for identifiability in practice
X <- matrix(rnorm(N*S), nrow = N, ncol = S)
X <- fpca.face(X)$Yhat ## smooth each curve using fast sandwich smoother
Z <- rnorm(N, 1, 1) ## a scalar predictor

data_analysis <- data.frame(event, survtime, Z, X = I(X)) ## the simulated dataset
rm(event, survtime, X, Z)

str(data_analysis)

## 'data.frame':    2000 obs. of  4 variables:
##  $ event      : int  0 1 1 0 0 1 0 0 1 1 ...
##  $ survtime: num  7.9091 6.2296 5.396 1.3632 0.0956 ...
##  $ Z          : num  1.157 1.413 0.622 0.538 -0.266 ...
```

```
## $ X      : 'AsIs' num [1:2000, 1:1000] 0.068 -0.0205 0.0136 -0.0466 -0.0266 ...
## ..- attr(*, "dimnames")=List of 2
## .. ..$ : NULL
## .. ..$ : NULL
```

One great feature of the `mgcv` package is that it allows to fit functional regression models using `gam()` function; see Chapter 7.11 of Wood (2017). To use this feature, we specify the covariates and the value of `by` parameter in smooth term functions (`s()`, `te()`, `ti()`) of `mgcv` as **matrices**. In practice, the function term is replaced by a discrete sum approximation.

$$\int_S \gamma(s) X_i(s) ds \approx \frac{1}{S} \sum_{k=1}^S \gamma(s_k) X_i(s_k).$$

We first create two variables related to such numerical approximation.

```
## create variables related to numerical approximation
### lmat: numerical integration
data_analysis$lmat <- I(matrix(1/S, ncol=S, nrow=nrow(data_analysis)))
### tmat: time indices of functional observations, we assume an equally-spaced grid here
data_analysis$tmat <- I(matrix(seq(0, 1, len=S), ncol=S, nrow=nrow(data_analysis), byrow=TRUE))
```

Following the instructions in `mgcv`, we next fit LFCM and AFCM using the `gam()` function.

```
## fit LFCM
fit_lfcm <- gam(survtime ~ Z + s(tmat, by=lmat*X, bs="cr", k=10), weights=event,
               data=data_analysis, family=cox.ph())

## fit AFCM
fit_afcm <- gam(survtime ~ Z + ti(tmat, X, by=lmat, bs=c("cr","cr"), k=c(10,10),
                                mc=c(FALSE,TRUE)), weights=event, data=data_analysis,
               family=cox.ph())
```

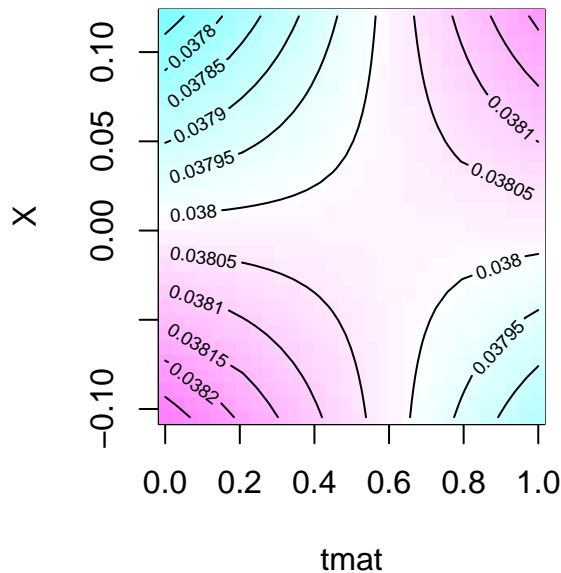
To fit a LFCM, we use the univariate smooth function `s()` to specify the linear functional term. Both `tmat` and `lmat*X` are matrices. In this example we use the cubic regression splines (`bs="cr"`) with 10 knots (`k=10`). The time to event variable, `survtime`, is set as the response in the formula, while the event indicator, `event`, is specified as the value of `weights` parameter following `mgcv`'s instructions.

To fit an AFCM, we use the tensor product function `ti()` to specify the bivariate additive term. Different from `te()` function, the `mc` parameter in `ti()` function allows to specify marginal centering constraints. When setting `mc=TRUE` for `X` direction, it exactly matches the additional identifiability constraints for AFCM.

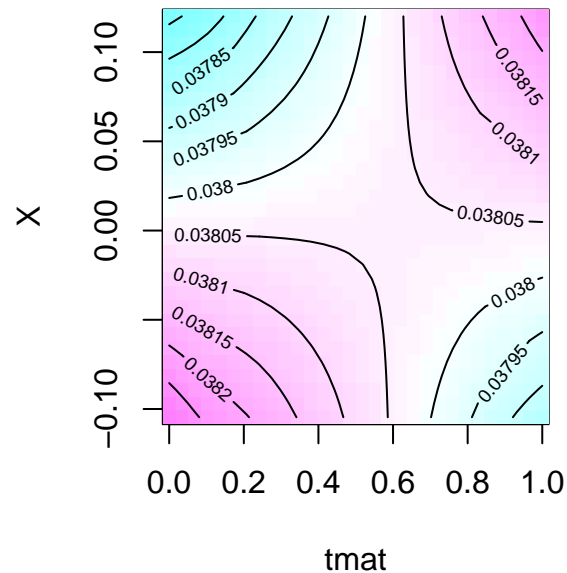
To visualize the estimates, one quick way is to use the `vis.gam()` function from `mgcv` package.

```
## visualize the estimates
par(mfrow = c(1,2))
vis.gam(fit_lfcm, view = c("tmat", "X"), plot.type = "contour", color = "cm",
        main = "Estimated Surface from LFCM")
vis.gam(fit_afcm, view = c("tmat", "X"), plot.type = "contour", color = "cm",
        main = "Estimated Surface from AFCM")
```

Estimated Surface from LFCM



Estimated Surface from AFCM



Part 2: NHANES

We now apply LFCM and AFCM to a real NHANES dataset stored in `NHANES.rds`. The detailed inclusion criteria can be found in Cui, Crainiceanu, and Leroux (2020).

```
## load dataset
data_analysis <- readRDS("./NHANES.rds")
dim(data_analysis)

## [1] 2816 20
str(data_analysis)

## 'data.frame': 2816 obs. of 20 variables:
## $ SEQN : int 21009 21010 21015 21019 21020 21033 21039 21050 21051 21055 ...
## $ BMI_cat : Factor w/ 4 levels "Normal","Underweight",...: 4 3 3 4 1 4 4 3 4 3 ...
## $ Race : Factor w/ 5 levels "White","Mexican American",...: 1 1 1 2 4 4 1 1 1 2 ...
## $ Gender : Factor w/ 2 levels "Male","Female": 1 2 1 2 2 2 2 1 1 2 ...
## $ Diabetes : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
## $ CHF : Factor w/ 2 levels "No","Yes": 1 1 1 2 1 1 1 2 1 1 ...
## $ CHD : Factor w/ 2 levels "No","Yes": 1 1 2 2 1 1 1 2 1 1 ...
## $ Cancer : Factor w/ 2 levels "No","Yes": 1 1 2 1 1 2 1 1 1 1 ...
## $ Stroke : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
## $ MobilityProblem: Factor w/ 2 levels "No Difficulty",...: 1 1 2 1 1 1 2 2 1 2 ...
## $ SmokeCigs : Factor w/ 3 levels "Never","Former",...: 1 3 2 2 1 1 2 2 2 1 ...
## $ Age : num 56 52.8 83.9 50.6 55.6 ...
## $ Overall_health : Factor w/ 5 levels "Excellent","Very good",...: 2 3 3 3 4 3 2 4 2 3 ...
## $ PIR : Factor w/ 3 levels "[0,1)","[1,2.5)",...: 3 2 2 1 2 1 1 2 3 2 ...
## $ Employed : Factor w/ 4 levels "Employed: full time",...: 1 3 3 3 1 3 3 3 1 1 ...
## $ Education : Factor w/ 3 levels "Less than high school",...: 2 3 3 1 1 2 2 2 2 1 ...
## $ Alcohol : Factor w/ 4 levels "Moderate Drinker",...: 2 3 2 2 2 2 2 2 2 2 ...
## $ time_mort : num 11.2 12.4 2 12.8 12.8 ...
## $ event : int 0 0 1 0 0 0 0 1 0 0 ...
```

```
## $ act_log_mat      : 'AsIs' num [1:2816, 1:1440] 0 0 0.256 0 0 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2816] "21009" "21010" "21015" "21019" ...
##   .. ..$ : chr [1:1440] "1" "2" "3" "4" ...
```

Among all variables, `act_log_mat` stores the minute-level LAC as a 2816×1440 matrix. The event indicator and time-to-event variables are stored in `event` and `time_mort`, respectively.

We first truncate the survival time at 10 years and smooth LAC. The smoothed LAC is our functional predictor $X_i(s)$ in this application.

```
## truncate time to event at 10 years
data_analysis$event[which(data_analysis$time_mort > 10)] <- 0
data_analysis$time_mort[which(data_analysis$time_mort > 10)] <- 10
table(data_analysis$event)

##
##      0      1
## 2157  659
```

```
## obtain smoothed LAC
data_analysis$act_log_mat_sm <- I(fPCA.face(unclass(data_analysis$act_log_mat))$Yhat)
```

To ensure estimability, we perform quantile transformation on the functional predictor.

```
## obtain quantile-transformed smoothed LAC
data_analysis$act_log_mat_sm_q <- I(apply(data_analysis$act_log_mat_sm, 2,
                                          function(y) ecdf(y)(y)))
```

We next fit LFCM and AFCM on **quantile-transformed** functional predictors using all steps introduced in Part 1. Directly fitting the models using untransformed data will yield uninterpretable patterns, as discussed in Cui, Crainiceanu, and Leroux (2020).

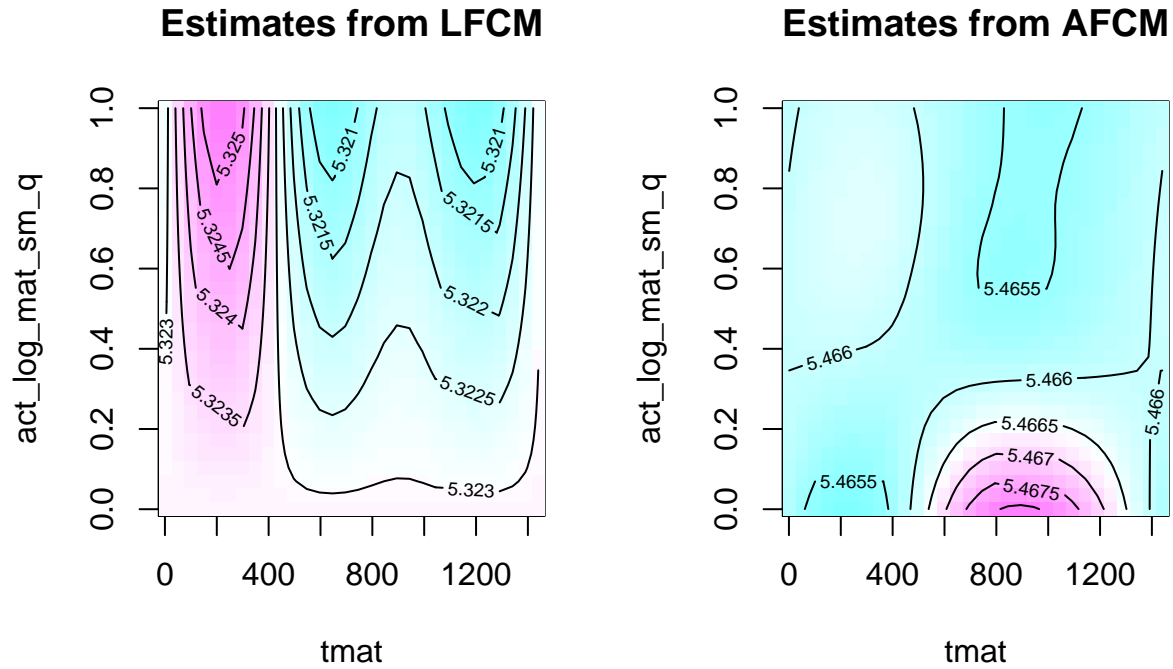
```
### lmat: numerical integration
data_analysis$lmat <- I(matrix(1/1440, ncol=1440, nrow=nrow(data_analysis)))
### tmat: time indices, [0, 1] in our case, but effectively arbitrary
data_analysis$tmat <- I(matrix(1:1440, ncol=1440, nrow=nrow(data_analysis), byrow=TRUE))

## fit LFCM
fit_lfcm <- gam(time_mort ~ Alcohol + Overall_health + PIR + Employed +
               Age + BMI_cat + SmokeCigs + Race + Education +
               CHD + Diabetes + CHF + Stroke + MobilityProblem + Cancer +
               s(tmat, by=lmat*act_log_mat_sm_q, bs="cc", k=10),
               weights=event, data=data_analysis, family=cox.ph())

## fit AFCM
fit_afcm <- gam(time_mort ~ Alcohol + Overall_health + PIR + Employed +
               Age + BMI_cat + SmokeCigs + Race + Education +
               CHD + Diabetes + CHF + Stroke + MobilityProblem + Cancer +
               ti(tmat, act_log_mat_sm_q, by=lmat, bs=c("cc","cr"),
                  k=c(5,5), mc=c(FALSE,TRUE)),
               weights=event, data=data_analysis, family=cox.ph())

## visualize the estimates
par(mfrow = c(1,2))
vis.gam(fit_lfcm, view = c("tmat", "act_log_mat_sm_q"), plot.type = "contour", color = "cm",
        main = "Estimates from LFCM")
vis.gam(fit_afcm, view = c("tmat", "act_log_mat_sm_q"), plot.type = "contour", color = "cm",
```

```
main = "Estimates from AFCM")
```



References

- Cui, Erjia, Ciprian M Crainiceanu, and Andrew Leroux. 2020. "Additive Functional Cox Model." *Journal of Computational and Graphical Statistics*, 1–14.
- Gellar, Jonathan E, Elizabeth Colantuoni, Dale M Needham, and Ciprian M Crainiceanu. 2015. "Cox Regression Models with Functional Covariates for Survival Data." *Statistical Modelling* 15 (3): 256–78.
- Kong, Dehan, Joseph G Ibrahim, Eunjee Lee, and Hongtu Zhu. 2018. "FLCRM: Functional Linear Cox Regression Model." *Biometrics* 74 (1): 109–17.
- Leroux, Andrew, Junrui Di, Ekaterina Smirnova, Elizabeth J McGuffey, Quy Cao, Elham Bayatmokhtari, Lucia Tabacu, Vadim Zipunnikov, Jacek K Urbanek, and Ciprian Crainiceanu. 2019. "Organizing and Analyzing the Activity Data in NHANES." *Statistics in Biosciences* 11 (2): 262–87.
- Wood, Simon N. 2017. *Generalized Additive Models: An Introduction with r*. CRC press.