# lecture_8_answer_key.R

Goodgolden5

2021-04-10

```
rm(list=ls())
library("tidyr");library("here"); library("readr"); library("dplyr");library("refund");library("mgcv")
```

```
## Warning: package 'tidyr' was built under R version 4.0.4
```

```
## here() starts at C:/Users/Goodgolden5/Desktop/BIOS7720/bios7720_functional
```

```
## Warning: package 'dplyr' was built under R version 4.0.4
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
## Warning: package 'refund' was built under R version 4.0.4
```

```
## Loading required package: nlme
```

```
##
## Attaching package: 'nlme'
```

```
## The following object is masked from 'package:dplyr':
##
##     collapse
```

```
## This is mgcv 1.8-33. For overview type 'help("mgcv-package")'.
```

```
#####################
##                 ##
## Data processing ##
##                 ##
#####################

## change data_path to where you have your data downloaded
data      <- read_rds(here("NHANES_AC_processed.rds"))
data_mort <- read_rds(here("data_mort.rds"))
## subset the data
data <-
    data %>%
    ## only consider good days of data and indiviudals age 50 or over
    filter(good_day %in% 1, Age > 50)
## get mortality data from the rnhanesdata package
```

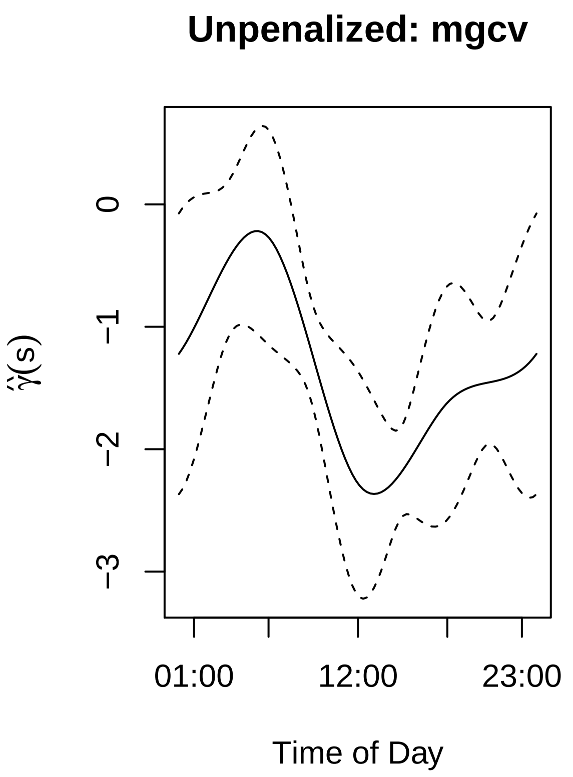
```r
## merge with our data and derive 5-year mortality indicator
data     <-
    left_join(data, data_mort, by="SEQN") %>%
    mutate(mort_5yr = as.numeric(permth_exm/12 <= 5 & mortstat %in% 1),
           ## replace accidental deaths within 5 years as NA
           mort_5yr = ifelse(mort_5yr == 1 & ucod_leading %in% "004", NA, mort_5yr)) %>%
    ## drop anyone missing mortality data or who had accidental deaths within 5 years
    filter(!is.na(mort_5yr))


## extract just the activity count data
Z <- log(as.matrix(data[,paste0("MIN",1:1440)])+1)
Z[is.na(Z)] <- 0
## average across days within participants (SEQN)
uid <- unique(data$SEQN)      # unique subject identifiers
nid <- length(uid)            # number of participants
Zmat <- matrix(NA, nid, 1440) # empty container to store average profiles
inx_ls <- lapply(uid, function(x) which(data$SEQN %in% x)) # list of indices
for(i in seq_along(uid)){
    Zmat[i,] <- colMeans(Z[inx_ls[[i]],,drop=FALSE])
}
## do fpca on the log(1+AC)
fpca_Z <- fpca.face(Y=Zmat, knots=50)
Zsm    <- fpca_Z$Yhat


## Get a data frame for analysis which contains one row per participant
df <- data[!duplicated(data$SEQN), ]
## drop the activity count columns
df <-
    df %>%
    dplyr::select(-one_of(paste0("MIN",1:1440)))
## add in the activity count matrix using the AsIs class via I()
## note!! be careful when working with dataframes which contain matrixes
df$Zsm  <- I(Zsm)
df$Zraw <- I(Zmat)
## clean up the workspace a bit
rm(Zsm);rm(Zmat);rm(Z)


## set up the functional domain matrix
## mgcv will use this to construct the basis \phi_k^\gamma(s)
sind  <- seq(0,1,len=1440)
smat  <- matrix(sind, nrow(df), 1440, byrow=TRUE)
df$smat <- I(smat)
## set up the matrix of integration weights
df$lmat <- I(matrix(1/1440, nrow(df), 1440))
## multiply integration weights by the functional predictor
df$zlmat <- I(df$lmat*df$Zsm)
fit_fglm_ps  <- gam(mort_5yr ~ s(smat, by=zlmat, bs="cc",k=30), data=df,
                    method="REML", family=binomial)
```

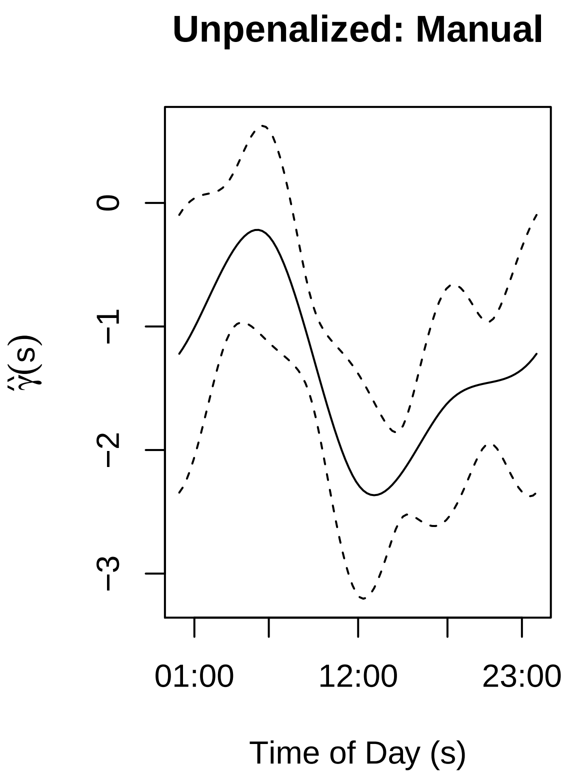
```r
#################
##             ##
##  Problem 1  ##
##             ##
#################

## Here we re-formulate the unpenalized GFLM as a linear regression problem

## Step 1: create basis functions and penalty matrix for
##         \gamma(s) using a small number (K=5) basis functions

## basis functions for the unpenalized fit
K_up <- 5
## create our matrix of basis functions on the functional domain S
## \phi_k(s)
smZ_up <- smoothCon(s(sind, bs="cc",k=K_up), data=data.frame(sind=sind))
## corresponding penalty matrix
# SZ  <- smZ_v1[[1]]$S[[1]]
## extract basis matrix on functional domain s
PhiS_up <- smZ_up[[1]]$X

## The integral term can be expressed as
##  zlmat * Phi
## Denote this quantity syntactically as "bigZ_up"
## where zlmat containes the elementwise product of the functional
## predictor and the quadrature weight at each observed point, assuming
## the functions are observed on the same grid (which we have here)

## create the integral term two ways
## 1) do the multiplication
## 2) loop over each subject

## first way
bigZ_up_v1 <- df$zlmat %*% PhiS_up

## second ("enefficient") way: looping over each individual
N <- nrow(df)
bigZ_up_v2 <- matrix(NA, N, K_up-1)
quad_vec <- rep(1/1440,1440)
tPhiS_up <- t(PhiS_up)
for(i in 1:N){
    bigZ_up_v2[i,] <- tPhiS_up %*% (df$Zsm[i,] * quad_vec)
}
## check to see they're the same by comparing MSE
## good!
mean((bigZ_up_v1 - bigZ_up_v2)^2)
```

```
## [1] 0
```

```r
## get the unpenalized model using our design matrix
gflm_fit_up_manual <- glm(mort_5yr~ bigZ_up_v1,family=binomial,data=df)
## get the unpenalized model using mgcv::gam
gflm_fit_up_mgcv   <- gam(mort_5yr~ s(smat, by=zlmat, bs="cc",k=5,fx=TRUE),family=binomial,data=df)
```

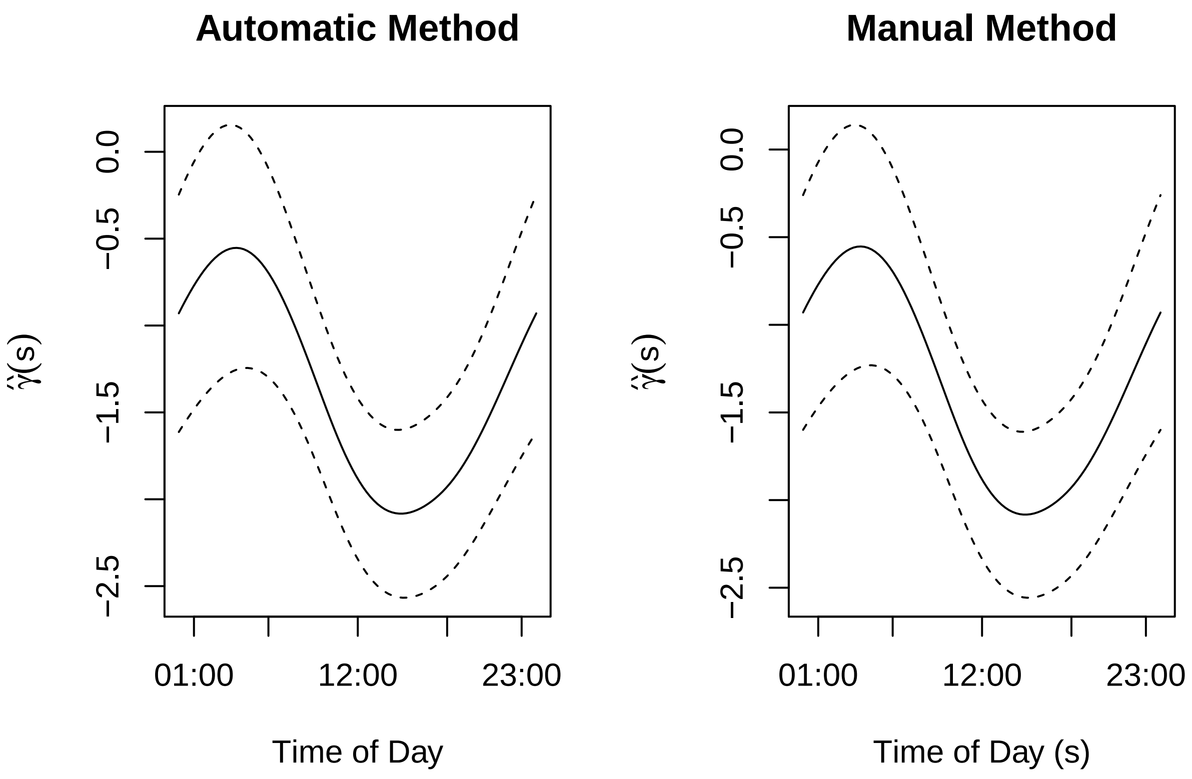
```
## get the estimated coefficients on a new grid of s values
sind_pred <- seq(0,1,len=100)
## get the basis matrix evaluated on the new s values
PhiS_up_pred <- PredictMat(smZ_up[[1]], data=data.frame(sind=sind_pred))
## evaluate \hat{gamma}(s) = Phi * \hat{xi}
## note that we exclude the first coefficient which is associated with the global
## intercept (\alpha_0)
fhat_up <- PhiS_up_pred %*% coef(gflm_fit_up_manual)[-1]
## get the standard errors using the formulat
## var(\hat{gamma}(s)) = var(Phi * \hat{xi}) = Phi var(\hat{xi}) Phi^t
## again we exclude the first row and column of the variance/covariance matrix
## because it's associated with the global intercept
se_fhat_up <- sqrt(diag(PhiS_up_pred %*% vcov(gflm_fit_up_manual)[-1,-1] %*% t(PhiS_up_pred)))

## plot the results
# x-axis ticks/labels
xinx <- (c(1,6,12,18,23)*60+1)/1440
xinx_lab <- c("01:00","06:00","12:00","18:00",'23:00')
# plot with two panels, 1 row and 2 columns
par(mfrow=c(1,2))
## plot the fit from mgcv::gam
plot(gflm_fit_up_mgcv, xlab="Time of Day", xaxt='n',
     ylab=expression(hat(gamma)(s)),main="Unpenalized: mgcv")
axis(1,at=xinx, xinx_lab)
## plot the fit usin our manual glm approach
matplot(sind_pred, cbind(fhat_up, fhat_up-1.96*se_fhat_up, fhat_up + 1.96*se_fhat_up),type='l',
        lty=c(1,2,2),col='black',xaxt='n',xlab="Time of Day (s)",main="Unpenalized: Manual",ylab=express
axis(1,at=xinx, xinx_lab)
```





```r
#################
##             ##
##  Problem 2  ##
##             ##
#################

# here we do the same thing only we use a greater number of basis functions
# since we'll be adding the penalization component which we specify using the "paraPen" argument
# as seen in the lecture slides

## basis functions for the penalized fit
K_pen <- 30
## create our matrix of basis functions on the functional domain S
smZ_pen  <- smoothCon(s(sind, bs="cc",k=K_pen), data=data.frame(sind=sind))
SZ       <- smZ_pen[[1]]$S[[1]]       # penalty matrix
PhiS_pen <- smZ_pen[[1]]$X            # basis matrix
bigZ_pen_v1 <- df$zlmat %*% PhiS_pen # integral term

## fit the manual penalized model
fit_pen_manual <- gam(mort_5yr ~ bigZ_pen_v1,
                      paraPen = list(bigZ_pen_v1=list(SZ)),
                      family=binomial,method="REML", data=df)
## fit the penalized model automatically
fit_pen_gam <- gam(mort_5yr ~ s(smat, by=zlmat, bs="cc",k=30),
                   data=df, method="REML", family=binomial)
## new grid of points to obtain estimates of \hat{f}(x) on
```

```
sind_pred           <- seq(0,1,len=100)
Phi_pred_pen_manual <- PredictMat(smZ_pen[[1]], data=data.frame(sind=sind_pred))
fhat_pen_manual     <- Phi_pred_pen_manual %*% coef(fit_pen_manual)[-1]
se_fhat_pen_manual  <- sqrt(diag(Phi_pred_pen_manual %*% vcov(fit_pen_manual)[-1,-1] %*% t(Phi_pred_pen_

## plot the results
par(mfrow=c(1,2))
# plot the fit from mgcv::gam
plot(fit_pen_gam, xlab="Time of Day", xaxt='n',ylab=expression(hat(gamma)(s)),main="Automatic Method")
axis(1,at=xinx, xinx_lab)
# plot the manual fit
matplot(sind_pred, cbind(fhat_pen_manual, fhat_pen_manual-1.96*se_fhat_pen_manual, fhat_pen_manual + 1.9
        lty=c(1,2,2),col='black',xaxt='n',xlab="Time of Day (s)",main="Manual Method",ylab=expression(ha
axis(1,at=xinx, xinx_lab)
```


Automatic Method
Manual Method
γ̂(s)
01:00
12:00
23:00
Time of Day
Time of Day (s)