R code for the dynamic prediction methods described in the paper: "Dynamic child growth prediction: a comparative methods approach"

Andrada E. Ivanescu and Ciprian Crainiceanu

In this document we present the R code corresponding to the BENDY, DLM, DPFR, and DPFFR dynamic prediction methods. Data is available for a time-dependent study variable, HAZ, and is stored in matrix format. We begin by describing the dataset structure. Data for HAZ is recorded as an $n \times 16$ matrix for n subjects where monthly data from month 0 until month 15 is available. This matrix is labeled Y. The i-th row, $1 \leq i \leq n$, denoted by `Y[i,]` contains HAZ data for the i-th subject in the sample.

Given t^* and available HAZ data, we provide the R code for predicting HAZ at $t^* + 1, t^* + 2, ...15$ using the dynamic prediction methods described in the paper "Dynamic child growth prediction: a comparative methods approach". The vector `month = 0:15` indicates a grid of equally spaced time points from month 0 until month 15 where HAZ data is available.

Throughout this persentation the case of interest is to predict future HAZ from the data corresponding to historic HAZ. Here t^* is labeled as `hist.lgth` to mark the length of known history for the observed HAZ process. We use leave one-curve out cross validation for prediction.

Libraries refund and mgcv are loaded.

```
library(refund)
library(mgcv)
```

Get the data and define number of samples $n = 197$, length of history $t^* = 7$, and vector `month = 0:15` for months $\{0, 1, 2, ..., 15\}$.

```
dta.y <- read.table("/Users/ai10/Desktop/HAZ_s.csv", sep = ",", header = FALSE)
Y = as.matrix(dta.y)
n = nrow(Y)

month = 0:15
hist.lgth <- 7
```

Initialize matrices where we store predictions.

```
Y.predict.BENDY <- matrix(nrow = n, ncol = length(month) - hist.lgth)
Y.predict.DLM <- matrix(nrow = n, ncol = length(month) - hist.lgth)
Y.predict.DPFR_gam <- matrix(nrow = n, ncol = length(month) - hist.lgth)
Y.predict.DPFR_pfr <- matrix(nrow = n, ncol = length(month) - hist.lgth)
Y.predict.DPFFR_gam <- matrix(nrow = n, ncol = length(month) - hist.lgth)
Y.predict.DPFFR_pffr <- matrix(nrow = n, ncol = length(month) - hist.lgth)
```

BENDY

Dynamic predictions with BENDY are obtained for each subject at each time point $t^* + j$. For all subjects except subject i we consider data `Y[-i,1]` and `Y[-i,t*]`, corresponding to first and last HAZ data from the known HAZ history, as predictive data for BENDY. We use all except the i-th subject to obtain the BENDY model fit, because we perform the leave one-curve out cross validation for prediction. The BENDY model fit is done n times to account for dynamic prediction for all n subjects.

The `lm` function in R is used to fit the BENDY model. For each dynamic prediction, a BENDY model fit is used. Model fit and prediction are needed for each j for dynamic prediction at times $t^* + j$.

```
for (i in 1:n) {
    for (j in 1:(length(month) - hist.lgth)) {
        data.BENDY <- data.frame(Y[-i, hist.lgth + j], Y[-i, 1], Y[-i, hist.lgth])
        names(data.BENDY) <- c("y.BENDY", paste("y", c(1, hist.lgth), sep = ""))
        fit.BENDY <- lm(y.BENDY ~ ., data = data.BENDY)
        new.data.BENDY <- data.frame(Y[i, 1], Y[i, hist.lgth])
        names(new.data.BENDY) <- c(paste("y", c(1, hist.lgth), sep = ""))
        Y.predict.BENDY[i, j] <- predict(fit.BENDY, newdata = new.data.BENDY)
    }
}
```

DLM

DLM uses more information than BENDY. While BENDY uses `Y[i,1]` and `Y[i,t*]` for dynamic prediction, DLM includes data `Y[i,1:t*]` which contains all the known history before time t^* for a subject i. The DLM model fit is done at each point $t^* + j$.

```
for (i in 1:n) {
    for (j in 1:(length(month) - hist.lgth)) {
        data.DLM <- data.frame(Y[-i, hist.lgth + j], cbind(Y[-i, 1:hist.lgth]))
        names(data.DLM) <- c("y.DLM", paste("y", c(1:hist.lgth), sep = ""))
        fit.DLM <- lm(y.DLM ~ ., data = data.DLM)
        new.data.DLM <- data.frame(rbind(Y[i, 1:hist.lgth]))
        names(new.data.DLM) <- c(paste("y", c(1:hist.lgth), sep = ""))
        Y.predict.DLM[i, j] <- predict(fit.DLM, newdata = new.data.DLM)
    }
}
```

DPFR

DPFR uses penalized functional regression to incorporate all the history of `Y` up to time t^* for subject i in a scalar-on-function regression. At each $t^* + j$ the response for subject i is the scalar `Y[i,t*+j]` and the functional predictor data consists of `Y[i,1:t*]`. We show how to obtain DPFR dynamic prediction using the function pfr from the refund R package and the gam function from the mgcv R package.

DPFR using pfr

DPFR can use pfr for model fitting. The function pfr from the refund R package directly takes on a scalar response `Y[i,t*+j]` and a functional predictor `Y[i,1:t*]`.

```
for(i in 1:n){
for(j in 1:(length(month)-hist.lgth)){
y.DPFR<-Y[-i,hist.lgth+j]
```

```
x.DPFR<-as.matrix(cbind(Y[-i,1:hist.lgth]))
fit.DPFR<-pfr(y.DPFR~lf(x.DPFR, k=4,bs="ps",
argvals=as.vector(cbind(month[1:hist.lgth]))))
Y.predict.DPFR_pfr[i,j]<-predict(fit.DPFR,newdata=list(x.DPFR=as.matrix(cbind(t(Y[i,1:hist.lgth])))),
type="response")
}}
```

DPFR using gam

Another option for fitting DPFR is using the gam function from the mgcv R package. To use gam, there is a step that involves arranging the data in some specific form prior to calling the gam function for DPFR model fitting.

```
for (i in 1:n) {
    for (j in 1:(length(month) - hist.lgth)) {
        y.s <- Y[-i, hist.lgth + j]
        X <- Y[-i, 1:hist.lgth]
        Lmat <- X[rep(1:(n - 1), each = 1), ]
        sngrid = hist.lgth
        smat <- matrix(month[1:hist.lgth], nrow = n - 1, nc = sngrid, byrow = TRUE)
        data.DPFR <- list()
        data.DPFR$y.s <- y.s
        data.DPFR$smat <- smat
        data.DPFR$Lmat <- Lmat
        fit.dpfr <- gam(y.s ~ s(smat, by = Lmat, bs = "ps", k = 4), data = data.DPFR,
            method = "REML")
        smat.new <- matrix(month[1:hist.lgth], nrow = 1, nc = sngrid, byrow = TRUE)
        X.new <- as.matrix(t(Y[i, 1:hist.lgth]), nrow = 1)
        Lmat.new <- t(X.new[rep(1:1, each = 1), ])
        data.DPFR.new <- list()
        data.DPFR.new$smat <- smat.new
        data.DPFR.new$Lmat <- Lmat.new
        DPFR.gam <- predict(fit.dpfr, data.DPFR.new, se = TRUE)
        Y.predict.DPFR_gam[i, j] <- DPFR.gam$fit
    }
}
```

DPFFR

DPFFR considers a functional response `Y[i,(t*+1):length(month)]` for each subject i. The predictor, `Y[i,1:t*]`, is also functional, which makes the approach a dynamic function-on-function regression. We provide the implementation of DPFFR with the pffr function from the refund R package and the gam function from the mgcv R package.

DPFFR using pffr

When using pffr from the refund R package, the user directly refers to the functional response `Y[i,(t*+1):length(month)]` and functional predictor `Y[i,1:t*]`. The response and predictor data are stored as matrices.

```
for (i in 1:n) {
    ymat <- Y[-i, (hist.lgth + 1):length(month)]
    X <- Y[-i, 1:hist.lgth]
    t.vec <- month[(hist.lgth + 1):length(month)]
    s.vec <- month[1:hist.lgth]
    data1 <- list()
    data1$ymat <- ymat
    data1$X <- X
    data1$t.vec <- t.vec
    data1$s.vec <- s.vec
    fit.DPFFR_pffr <- pffr(ymat ~ ff(X, xind = s.vec), yind = t.vec, data = data1)
    data2 <- list()
    data2$X <- as.matrix(t(Y[i, 1:hist.lgth]))
    Y.predict.DPFFR_pffr[i, ] <- predict(fit.DPFFR_pffr, newdata = data2)
}
```

DPFFR using gam

Using gam from the mgcv R package is an option for DPFFR. When using gam for DPFFR there is a required step of having the functional response `Y[i,(t*+1):length(month)]` in vector format. Additional steps are needed to generate the required data format for gam. Details are shown below.

```
for (i in 1:n) {
    Y.v <- Y[, (hist.lgth + 1):length(month)]
    X <- Y[, 1:hist.lgth]
    yvec <- as.vector(t(Y.v[-i, ]))
    t <- month[(hist.lgth + 1):length(month)]
    s <- month[1:hist.lgth]
    by = 1
    tngrid = length(t)
    sngrid = length(s)
    tmat <- matrix(t, nrow = (n - 1) * tngrid, nc = sngrid, byrow = FALSE)
    smat <- matrix(s, nrow = (n - 1) * tngrid, nc = sngrid, byrow = TRUE)
    LX <- X[-i, ]
    Lmat <- LX[rep(1:(n - 1), each = tngrid), ]
    tvec <- matrix(t, nrow = (n - 1) * tngrid, nc = 1, byrow = FALSE)
    data.DPFFR <- list()
    data.DPFFR$yvec <- yvec
    data.DPFFR$tvec <- tvec
    data.DPFFR$tmat <- tmat
    data.DPFFR$smat <- smat
    data.DPFFR$Lmat <- Lmat
    fit.DPFFR <- gam(yvec ~ s(tvec, bs = "ps", k = 4) + te(tmat, smat, by = Lmat,
        bs = "ps"), method = "REML")
    tmat.new <- matrix(t, nrow = (1) * tngrid, nc = sngrid, byrow = FALSE)
    smat.new <- matrix(s, nrow = (1) * tngrid, nc = sngrid, byrow = TRUE)
    L <- matrix(by, ncol = length(s), nrow = 1)
    LX.new <- L * X[i, ]
    Lmat.new <- LX.new[rep(1:(1), each = tngrid), ]
    tvec.new <- matrix(t, nrow = (1) * tngrid, nc = 1, byrow = FALSE)
    data.DPFFR.new <- list()
    data.DPFFR.new$tvec <- tvec.new
```

```
    data.DPFFR.new$tmat <- tmat.new
    data.DPFFR.new$smat <- smat.new
    data.DPFFR.new$Lmat <- Lmat.new
    predict.GAM <- predict(fit.DPFFR, data.DPFFR.new, se = TRUE)
    Y.predict.DPFFR_gam[i, ] <- predict.GAM$fit
}
```

Dynamic Prediction Graphics

Below we provide the results for dynamic prediction for a subject. Dynamic predictions are illustrated for each method. In each graph the red solid line represents the dynamic prediction. Historic data is shown as a blue solid line.