BIOS 7720: Applied Functional Data Analysis

Lecture 13: Multilevel fPCA

Andrew Leroux

May 6, 2021

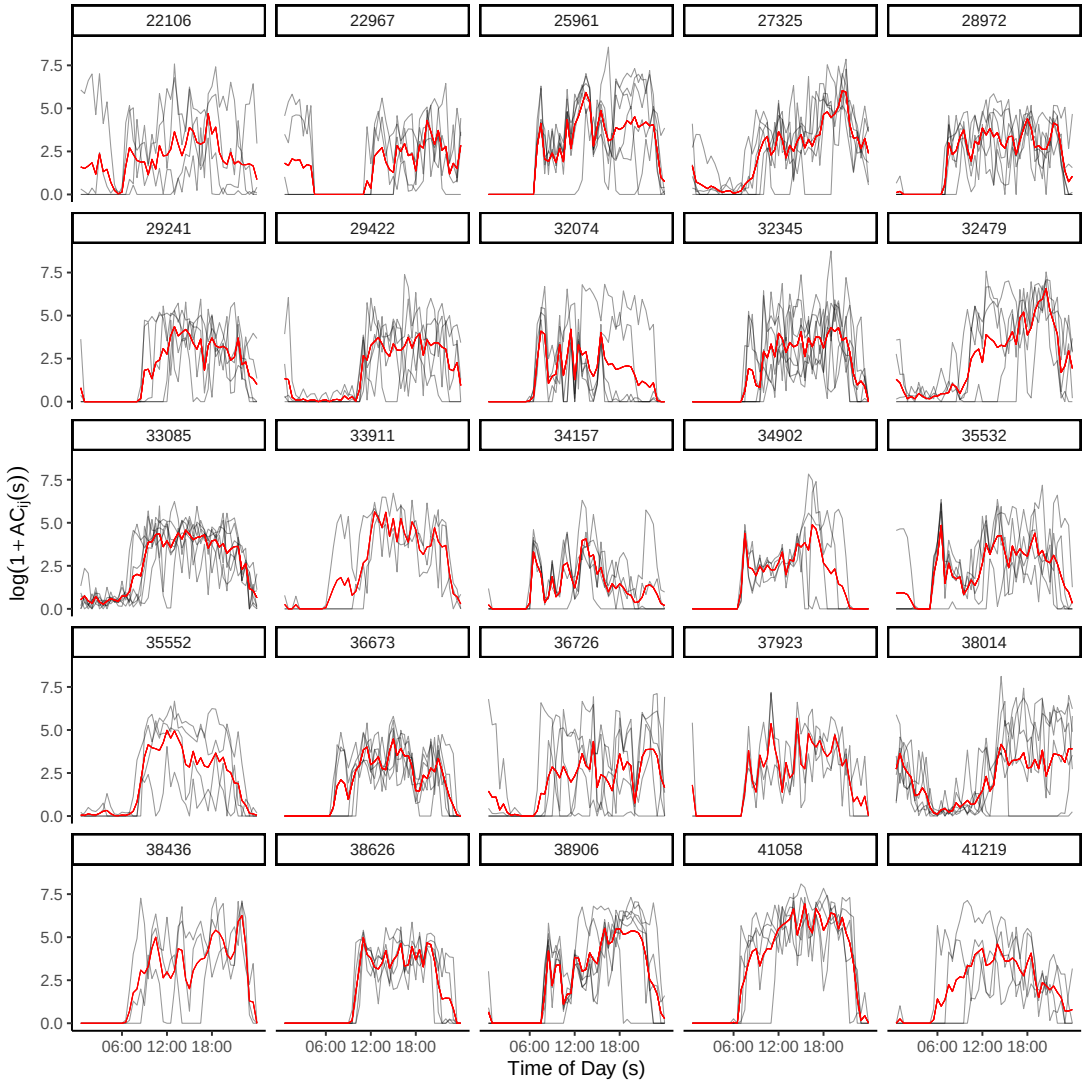
Logistics

- HW 3 due 5/13
- Final group projects
  - Presentations 5/11 and 5/13
  - Write up due 5/20
- Group presentation order

Multilevel Functional Data

```r
## change data_path to whereever your NHANES file is located
data_path <- here::here("data","data_processed","NHANES_AC_processed.rds")
df        <- readr::read_rds(data_path)
df_sub    <-
  df %>%
  filter(n_good_days >= 3, good_day == 1,  Age <= 25)
uid <- unique(df_sub$SEQN)

## extract the PA data
lX <- log(1+as.matrix(df_sub[,paste0("MIN",1:1440)]))
lX[is.na(lX)] <- 0
N  <- nrow(lX)
## bin the data into 30 minute intervals
tlen <- 30
nt   <- ceiling(1440/tlen)
inx_cols <- split(1:1440, rep(1:nt, each=tlen)[1:1440])
lX_bin   <- vapply(inx_cols, function(x) rowMeans(lX[,x], na.rm=TRUE), numeric(N))
colnames(lX_bin) <- paste0("epoch_",1:nt)
```

Multilevel Functional Data



Multilevel fPCA

- Multilevel fPCA [Di et al., 2009]

$$
\begin{aligned}
Y_{ij}(s) &= \mu_0(s) + b_i(s) + \nu_{ij}(s) + \epsilon_{ij}(s) \\
b_i &\overset{\text{iid}}{\sim} GP(0, \boldsymbol{\Sigma_b}) \\
\nu_{ij} &\overset{\text{iid}}{\sim} GP(0, \boldsymbol{\Sigma_\nu}) \\
\epsilon_{ij}(s) &\overset{\text{iid}}{\sim} N(0, \sigma^2_\epsilon)
\end{aligned}
$$

- b_i is the participant deviation from the population average
- ν_{ij} is the day deviation from a participants' average $(\mu_0(s) + b_i(s))$

Multilevel Functional Data

```
set.seed(10)
## remove unnecessary columns
df_sub <-
  df_sub %>%
  dplyr::select(-one_of(paste0("MIN",1:1440)))
## add in the binned data to our data frame
df_sub[["lX_bin"]] <- lX_bin
## subset to only 100 participants
nsamp <- 100
uid_samp <- sample(uid, size=nsamp, replace=FALSE)
df_mfpca <-
  filter(df_sub, SEQN %in% uid_samp) %>%
  group_by(SEQN) %>%
  mutate(J = 1:n()) %>%
  ungroup() %>%
  arrange(SEQN, J)
```

Multilevel Functional Data

```
library("refund")
mfpca_fit <- mfpca.sc(Y=df_mfpca$lX_bin,
                      id=df_mfpca$SEQN,
                      visit=df_mfpca$J,
                      pve=0.95)
```

Multilevel Functional Data: Estimation

- Suppose we observe $y_{ij}(s)$ for
  - participant (level 1) $i = 1, \ldots, N$
  - visit (level 2) $j = 1, \ldots, J$
- We want to estimate the mfPCA model

$$y_{ij}(s) = \mu(s) + b_i(s) + \nu_{ij}(s) + \epsilon_{ij}(s)$$

Multilevel Functional Data: Estimation

- Suppose we observe $y_{ij}(s)$ for
  - participant (level 1) $i = 1, \ldots, N$
  - visit (level 2) $j = 1, \ldots, J$
- We want to estimate the mfPCA model

$$y_{ij}(s) = \mu(s) + b_i(s) + \nu_{ij}(s) + \epsilon_{ij}(s)$$

- Start with no error present

$$y_{ij}(s) = \mu(s) + b_i(s) + \nu_{ij}(s)$$

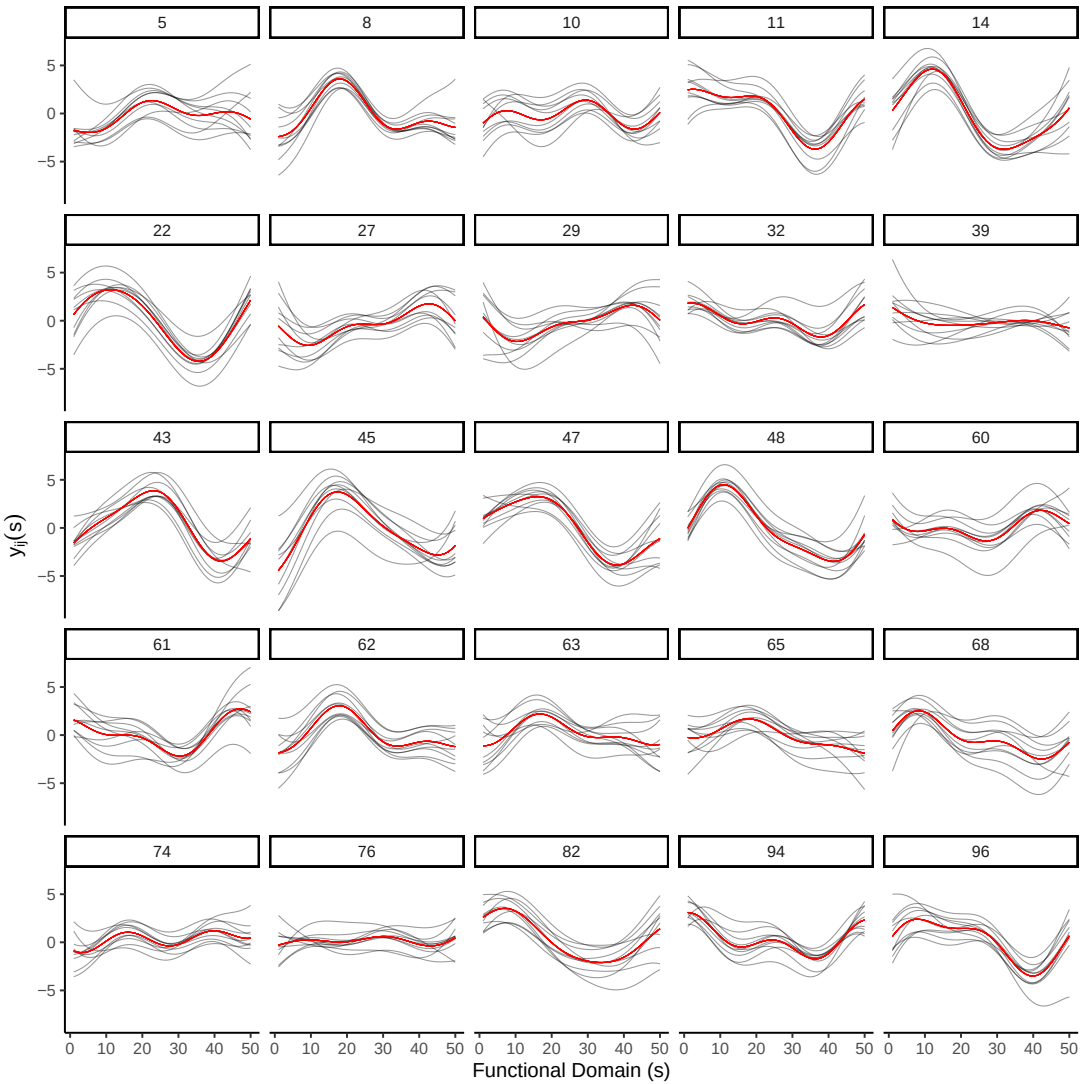
Multilevel Functional Data: Estimation

```r
set.seed(1989)
N <- 100 # number of subjects
J <- 10  # number of visits per subject
nS   <- 50 # number of observations of the functional domain
sind <- seq(0,1,len=nS)

# true eigenvalues
K1 <- K2 <- 4
lambda1 <- 0.5^(0:(K1-1)) # for lvl 1
lambda2 <- 0.5^(0:(K2-1)) # for lvl 2

# true Eigenfunctions
Phi1 <- sqrt(2)*cbind(sin(2*pi*sind),cos(2*pi*sind),
                      sin(4*pi*sind),cos(4*pi*sind))
Phi2 <- cbind(rep(1,nS),
              sqrt(3)*(2*sind-1),
              sqrt(5)*(6*sind^2-6*sind+1),
              sqrt(7)*(20*sind^3-30*sind^2+12*sind-1))
```

Multilevel Functional Data: Estimation

```
# simulate b_i(s)
xi_lvl1 <- matrix(rnorm(N*K1),N,K1);
xi_lvl1 <- xi_lvl1 %*% diag(sqrt(lambda1))
b_i <- xi_lvl1 %*% t(Phi1);

# simulate nu_ij(s)
xi_lvl2 <- matrix(rnorm(N*J*K2),N*J,K2);
xi_lvl2 <- xi_lvl2 %*% diag(sqrt(lambda2))
nu_ij <- xi_lvl2 %*% t(Phi2)

# combine to create y_ij(s)
mu_0 <- function(s) sin(2*pi*s)
b_i_mat <- kronecker(b_i,matrix(1, J, 1))
y_ij    <- kronecker(matrix(mu_0(sind),1,nS), matrix(1,N*J, 1)) +
                nu_ij +
                b_i_mat

df_mfpca_sim <- data.frame("id"=rep(1:N, each=J),
                           "visit" = rep(1:J, N),
                           "b_i" = I(b_i_mat),
                           "y_ij" = I(y_ij),
                           "nu_ij" = I(nu_ij))
```

Multilevel Functional Data: Estimation



Multilevel Functional Data: Estimation

- We have three covariances functions of interest

$$K_T(s,u) = \text{Cov}(y_{ij}(s), y_{ij}(u))$$
$$K_B(s,u) = \text{Cov}(y_{ij}(s), y_{ik}(u))$$
$$K_W(s,u) = K_T(s,u) - K_B(s,u)$$

- Follow a similar estimation procedure as fPCA
- Estimate a mean model, decompose covariance of the residuals (K_T)

Multilevel Functional Data: Estimation

$$
\begin{aligned}
K_T(s,u) &= \mathrm{Cov}(y_{ij}(s), y_{ij}(u)) \\
&= \mathrm{Cov}(\mu(s) + b_i(s) + \nu_{ij}(s), \mu(s) + b_i(s) + \nu_{ij}(s)) \\
&= \mathrm{Cov}(b_i(s) + \nu_{ij}(s), b_i(u) + \nu_{ij}(u)) \\
&= \mathrm{Cov}(b_i(s), b_i(u)) + \mathrm{Cov}(b_i(s), \nu_{ij}(u)) + \\
&\qquad \mathrm{Cov}(\nu_{ij}(s), b_i(u)) + \mathrm{Cov}(\nu_{ij}(s), \nu_{ij}(u)) \\
&= \mathrm{Cov}(b_i(s), b_i(u)) + \mathrm{Cov}(\nu_{ij}(s), \nu_{ij}(u)) \\
&= \Sigma_b + \Sigma_\nu \\
&= \sum_{k=1}^{\infty} \lambda_k^b \phi_k^b(s) \phi_k^b(u) + \sum_{k=1}^{\infty} \lambda_k^\nu \phi_k^\nu(s) \phi_k^\nu(u) \\
&\approx \sum_{k=1}^{K_b} \lambda_k^b \phi_k^b(s) \phi_k^b(u) + \sum_{k=1}^{K_\nu} \lambda_k^\nu \phi_k^\nu(s) \phi_k^\nu(u)
\end{aligned}
$$

Multilevel Functional Data: Estimation

$$
\begin{aligned}
K_B(s,u) &= \mathrm{Cov}(y_{ij}(s), y_{ik}(u)) \\
&= \mathrm{Cov}(\mu(s) + b_i(s) + \nu_{ij}(s), \mu(s) + b_i(s) + \nu_{ik}(s)) \\
&= \mathrm{Cov}(b_i(s) + \nu_{ij}(s), b_i(u) + \nu_{ik}(u)) \\
&= \mathrm{Cov}(b_i(s), b_i(u)) + \mathrm{Cov}(b_i(s), \nu_{ik}(u)) + \\
&\qquad \mathrm{Cov}(\nu_{ij}(s), b_i(u)) + \mathrm{Cov}(\nu_{ik}(s), \nu_{ij}(u)) \\
&= \mathrm{Cov}(b_i(s), b_i(u)) \\
&= \Sigma_b \\
&= \sum_{k=1}^{\infty} \lambda_k^b \phi_k^b(s) \phi_k^b(u) \\
&\approx \sum_{k=1}^{K_b} \lambda_k^b \phi_k^b(s) \phi_k^b(u)
\end{aligned}
$$

Multilevel Functional Data: Estimation

$$\hat{K}_T(s,u) = \frac{\sum_{i=1}^{N}\sum_{j=1}^{J}(y_{ij}(s)-\hat{\mu}_0(s))(y_{ij}(u)-\hat{\mu}_0(u))}{(N\times J)}$$

$$\hat{K}_B(s,u) = \frac{\sum_{i=1}^{N}\sum_{j_1\neq j_2}(y_{ij_1}(s)-\hat{\mu}_0(s))(y_{ij_2}(u)-\hat{\mu}_0(u))}{(N\times J\times (J-1))}$$

$$\hat{K}_W(s,u) = \hat{K}_T(s,u) - \hat{K}_B(s,u)$$

Multilevel Functional Data: Estimation

```
# library("mgcv")
## set up the long data frame for estimating mean function
## (could also use empirical mean here since data are measured without error)
# df_mfpca_long <-
#   data.frame("id" = rep(df_mfpca$id, each=nS),
#              "visit" = rep(df_mfpca$visit, each=nS),
#              "y" = as.vector(t(df_mfpca$y_ij)),
#              "sind" = rep(sind, N*J))
# fit_naive <- gam(y ~ s(sind, bs="cr",k=30), method="REML", data=df_mfpca_long)
# resid_mat <- matrix(fit_naive$residuals, N*J, nS, byrow=TRUE)
resid_mat <- y_ij - kronecker(matrix(colMeans(y_ij),1,nS), matrix(1, N*J, 1))
KT_hat <- KB_hat <- matrix(0, nS, nS)

## estimate K_T
inx <- 1
for(i in 1:N){
  for(j in 1:J){
    KT_hat <- KT_hat + tcrossprod(resid_mat[inx,])
    inx <- inx+1
  }
}
```

Multilevel Functional Data: Estimation

```
## estimate K_B
inx_i <- 1:J
for(i in 1:N){
  for(j1 in 1:(J-1)){
    for(j2 in (j1+1):J){
      KB_hat = KB_hat +
                tcrossprod(resid_mat[inx_i[j1],], resid_mat[inx_i[j2],]) +
                tcrossprod(resid_mat[inx_i[j2],], resid_mat[inx_i[j1],])
    }
  }
  inx_i <- inx_i + J
}
KT_hat <- KT_hat / (N*J)
KB_hat <- KB_hat / (N*J*(J-1))
KW_hat <- KT_hat - KB_hat
```

Multilevel Functional Data: Estimation

```
matrixcalc::is.positive.semi.definite(KT_hat)

## [1] TRUE

matrixcalc::is.positive.semi.definite(KB_hat)

## Error in matrixcalc::is.positive.semi.definite(KB_hat):  argument x is not
a symmetric matrix

matrixcalc::is.positive.semi.definite(KW_hat)

## Error in matrixcalc::is.positive.semi.definite(KW_hat):  argument x is not
a symmetric matrix
```

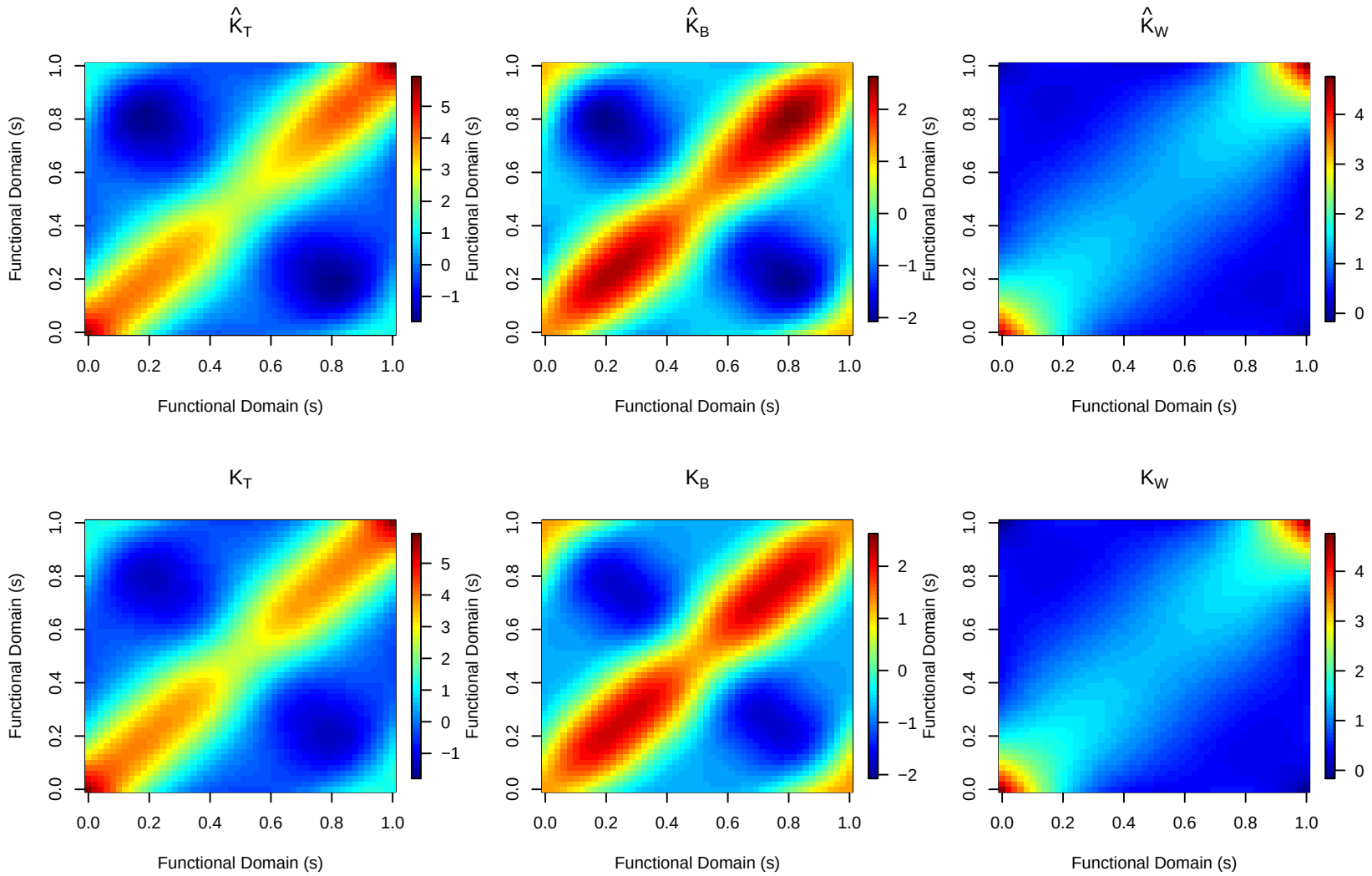
Multilevel Functional Data: Estimation

```
KB_hat[1:5,1:5]

##          [,1]     [,2]     [,3]     [,4]     [,5]
## [1,] 1.370842 1.337534 1.271967 1.175052 1.049277
## [2,] 1.337534 1.376581 1.381063 1.349283 1.281382
## [3,] 1.271967 1.381063 1.454678 1.488579 1.480541
## [4,] 1.175052 1.349283 1.488579 1.586367 1.638159
## [5,] 1.049277 1.281382 1.480541 1.638159 1.747694

matrixcalc::is.symmetric.matrix(round(KB_hat,12))

## [1] TRUE

matrixcalc::is.symmetric.matrix(round(KB_hat,13))

## [1] FALSE

matrixcalc::is.positive.semi.definite(round(KB_hat,12))

## [1] FALSE
```

Multilevel Functional Data: Estimation

```
## make K_B positive-semidefinite
eigen_KB <- eigen(KB_hat)
inx_rm_1 <- which(eigen_KB$values < 0)
if(length(inx_rm_1) > 0){
  eigen_KB$values[inx_rm_1] <- 0
  eigen_KB$vectors[,inx_rm_1] <- 0
}
KB_hat_trim <- eigen_KB$vectors %*% diag(eigen_KB$values) %*% t(eigen_KB$vectors)

## make K_W positive-semidefinite
KW_hat <- KT_hat-KB_hat_trim
eigen_KW <- eigen(KW_hat)
(inx_rm_2 <- which(eigen_KW$values < 0))

##  [1] 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43
## [26] 44 45 46 47 48 49 50

if(length(inx_rm_1) > 0){
  eigen_KW$values[inx_rm_2] <- 0
  eigen_KW$vectors[,inx_rm_2] <- 0
}
KW_hat_trim <- eigen_KW$vectors %*% diag(eigen_KW$values) %*% t(eigen_KW$vectors)
```

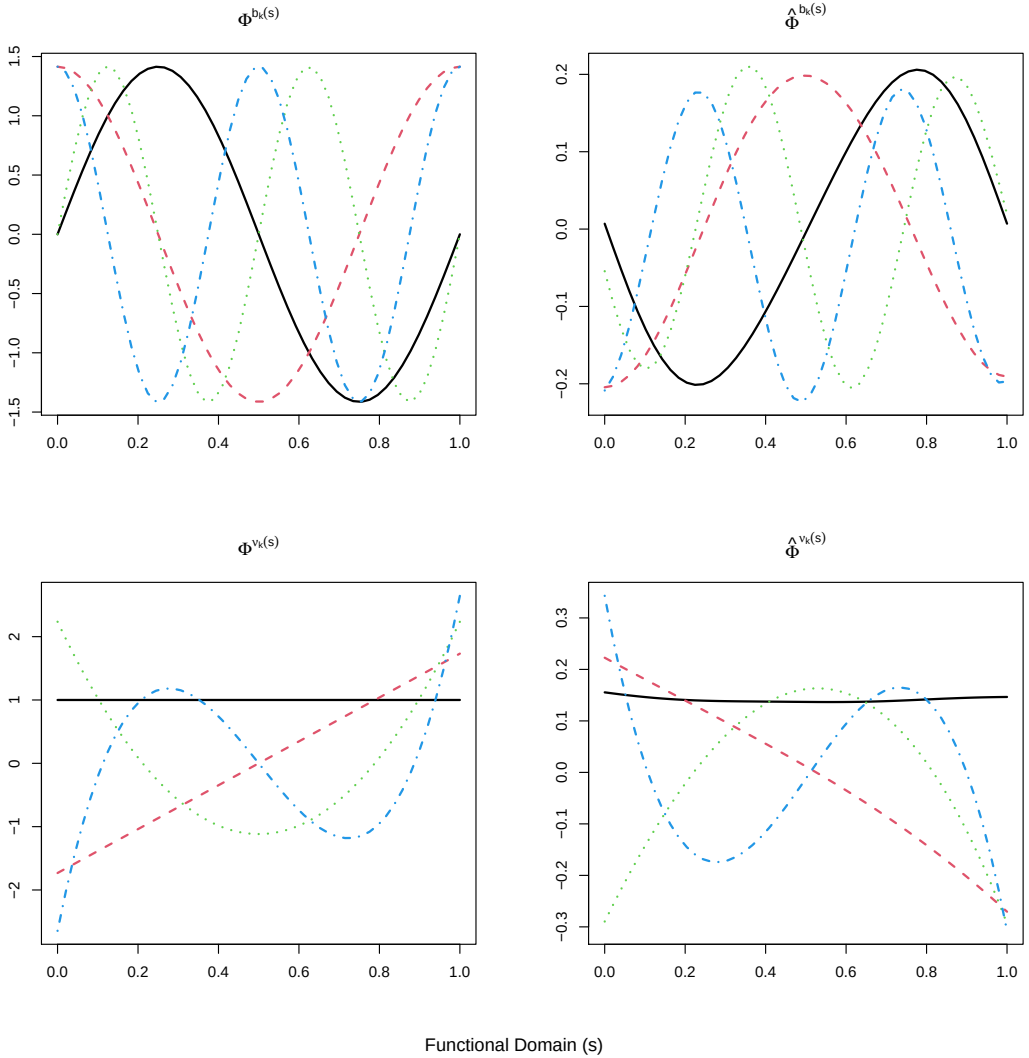
Multilevel Functional Data: Estimation



Multilevel Functional Data: Estimation

```
## choose the number of eigenvalues
thresh <- 0.95
(pve_lvl1 <- cumsum(eigen_KB$values)/sum(eigen_KB$values))[1:5]

## [1] 0.5527630 0.7905170 0.9242457 0.9970893 0.9999664

(pve_lvl2 <- cumsum(eigen_KW$values)/sum(eigen_KW$values))[1:5]

## [1] 0.5099347 0.7835731 0.9304907 1.0000000 1.0000000

K1 <- min(which(pve_lvl1 >= thresh))
K2 <- min(which(pve_lvl2 >= thresh))
## get the eigenvectors
Phi1_hat <- eigen_KB$vectors[,1:K1]
Phi2_hat <- eigen_KW$vectors[,1:K2]
## get the eigenvalues
lambda1_hat <- eigen_KB$values[1:K1]
lambda2_hat <- eigen_KW$values[1:K2]
```

Multilevel Functional Data: Estimation



Multilevel Functional Data: Prediction

$$\begin{bmatrix} \xi^b \\ \xi^\nu \end{bmatrix} = \begin{bmatrix} \boldsymbol{A_i} \\ \boldsymbol{B_i} \end{bmatrix} \boldsymbol{\Sigma}_i^{-1}(\boldsymbol{y}_i - \hat{\boldsymbol{\mu}}_i)$$

$$\boldsymbol{A}_i = 1_J^t \otimes (\boldsymbol{\Lambda}^b \boldsymbol{\Phi}^{b,t})$$

$$\boldsymbol{B}_i = \boldsymbol{I}_J \otimes (\boldsymbol{\Lambda}^\nu \boldsymbol{\Phi}^{\nu,t})$$

$$\boldsymbol{\Sigma}_i = \mathrm{Cov}(\boldsymbol{y}_i, \boldsymbol{y}_i)$$

Note that the math is a bit more complicated if not all participants are fully observed on the same grid

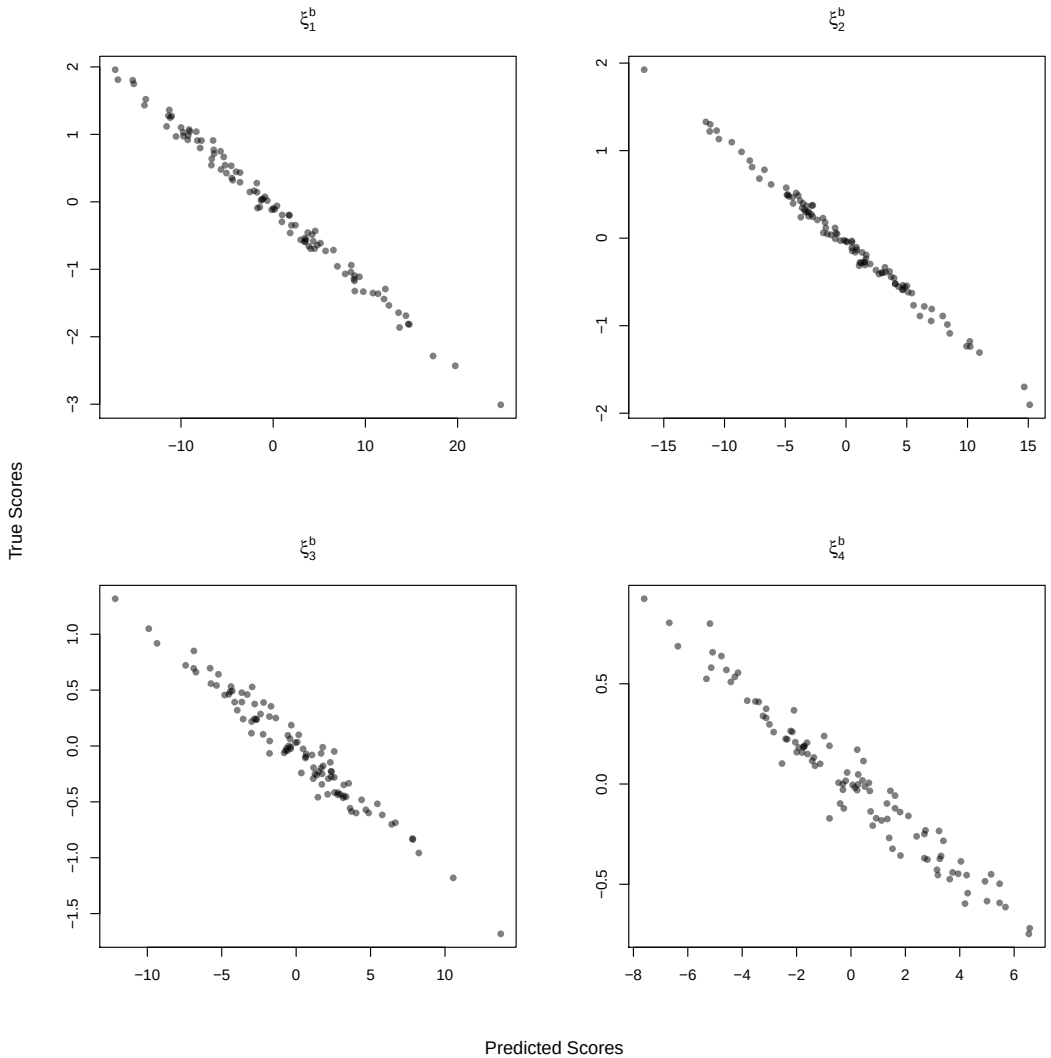
Multilevel Functional Data: Prediction

```
Lambda_b <- diag(lambda1_hat)
Lambda_nu <- diag(lambda2_hat)
A_i <- kronecker(matrix(1, 1, J),Lambda_b%*%  t(Phi1_hat))
B_i <-  kronecker(diag(1, J), Lambda_nu %*%t(Phi2_hat))
Sigma_ii <- KW_hat_trim + KB_hat_trim
Sigma_ij <- KB_hat_trim
Sigma_i <-  as.matrix(Matrix::bdiag(lapply(1:J, function(x) Sigma_ii)))
inx_j1 <- 1:nS
for(j1 in 1:J){
  if(j1 < J-1){
    for(j2 in (j1+1):(J-1)){
      inx_j2 <- 1:nS + nS*j2
      Sigma_i[inx_j1,inx_j2] <- Sigma_ij
      Sigma_i[inx_j2,inx_j1] <- t(Sigma_ij)
    }
  }
  inx_j1 <- inx_j1 + nS
}
```

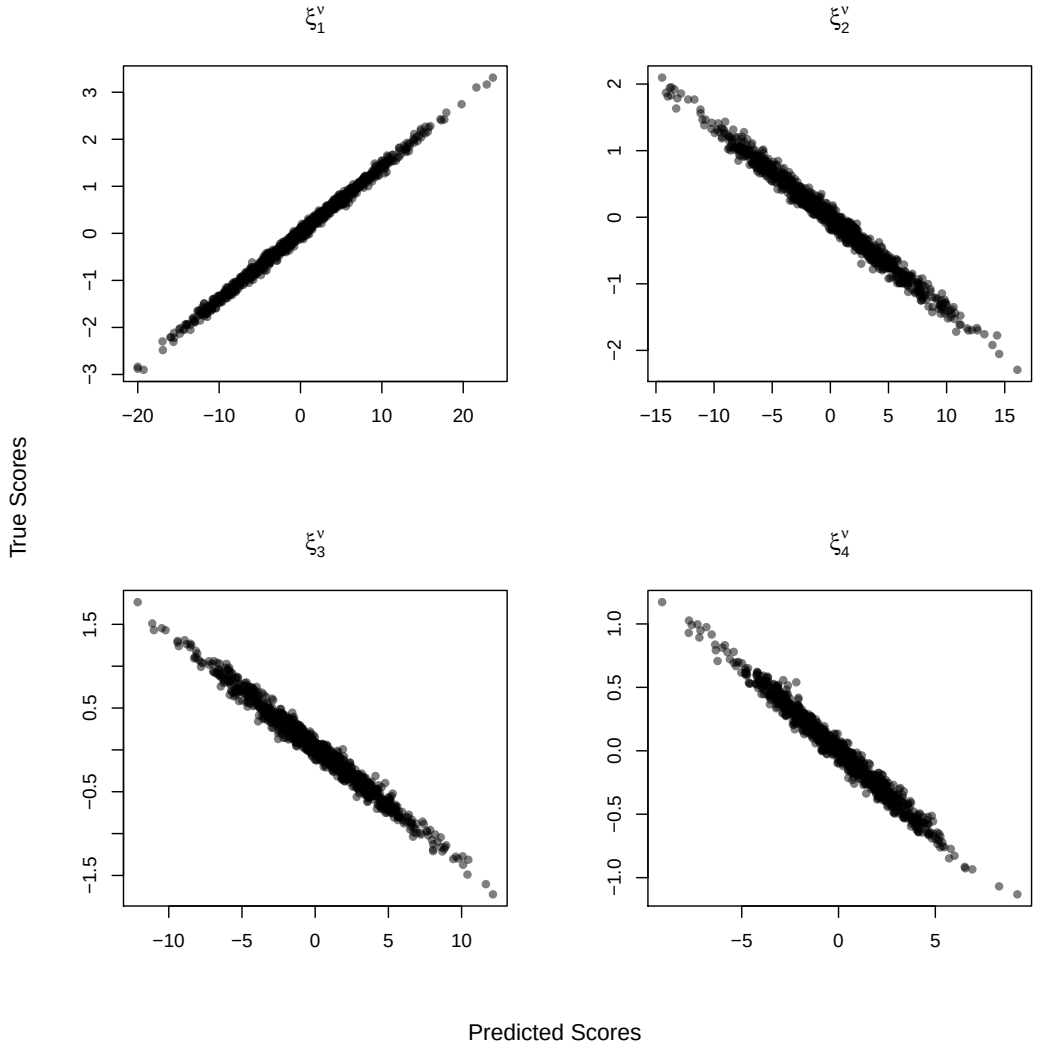
Multilevel Functional Data: Prediction

```
Sigma_i_inv <- MASS::ginv(Sigma_i)
inx_i <- 1:J
xi_hat_lvl1 <- matrix(NA, N, K1)
xi_hat_lvl2 <- matrix(NA, N*J, K2)
for(i in 1:N){
  r_i <- as.vector(t(resid_mat[inx_i,]))
  scores_i <- rbind(A_i, B_i) %*% Sigma_i_inv %*% r_i
  xi_hat_lvl1[i,] <- scores_i[1:K1]
  xi_hat_lvl2[inx_i,] <- matrix(scores_i[-c(1:K1)], J, K2, byrow=TRUE)
  inx_i <- inx_i + J
}
```

Multilevel Functional Data: Prediction



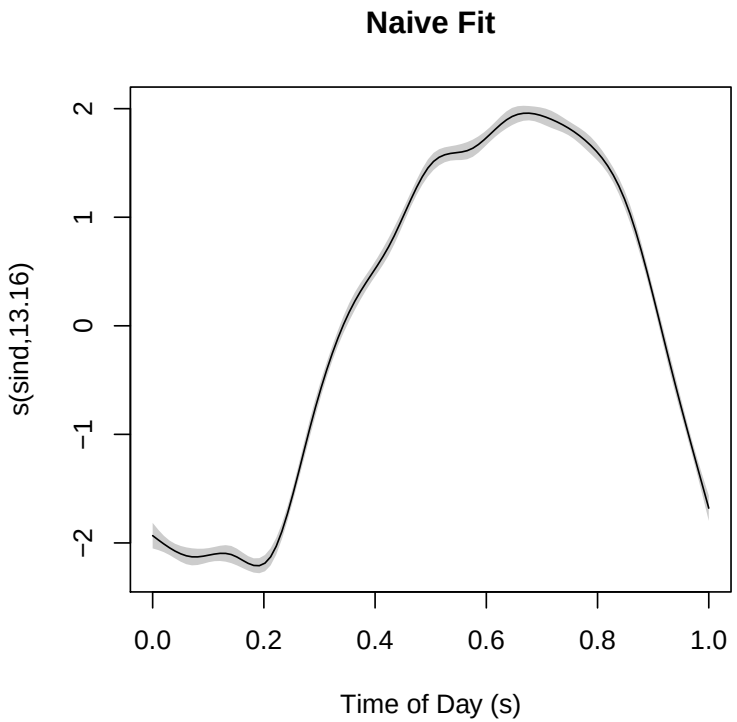
Multilevel Functional Data: Prediction

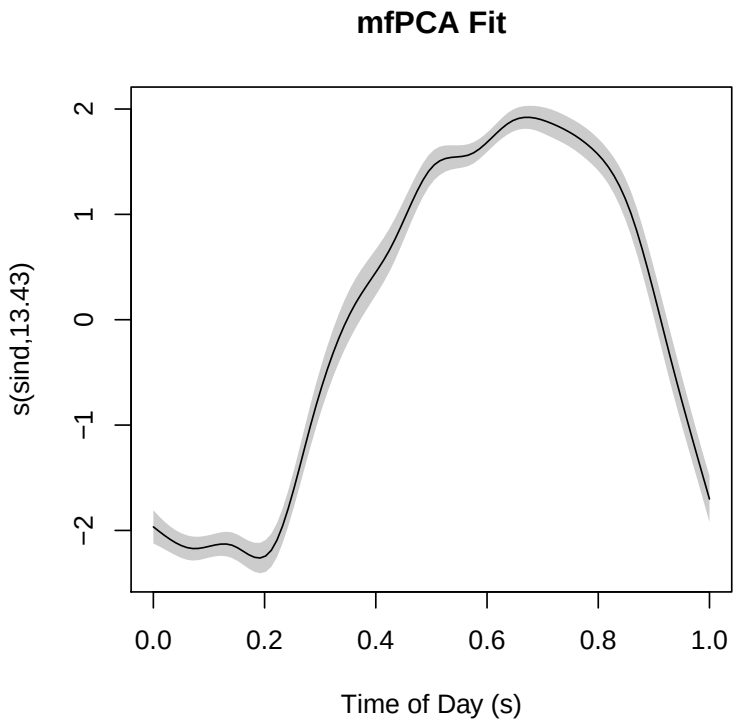


Multilevel Functional Data: *mgcv*

```
library("mgcv")
sind <- seq(0,1,len=nt)
N    <- nrow(df_mfpca)
df_mfpca_long <-
  data.frame("id" = factor(rep(df_mfpca$SEQN, each=nt)),
             "day" = rep(df_mfpca$J, each=nt),
             "sind" = rep(sind,N),
             "y" = as.vector(t(df_mfpca$lX_bin)))
df_mfpca_long$id_day = factor(paste0(df_mfpca_long$id,"_",df_mfpca_long$day))
K1 <- K2 <- 2
for(k in 1:K1){
  df_mfpca_long[[paste0("phi_b_",k)]] <-
    rep(mfpca_fit$efunctions$level1[,k],N)
}
for(k in 1:K2){
  df_mfpca_long[[paste0("phi_nu_",k)]] <-
    rep(mfpca_fit$efunctions$level2[,k],N)
}
fit_naive <-  bam(y ~ s(sind, bs="cr", k=15),
                  data=df_mfpca_long, method="fREML", discrete=TRUE)
fit_mgcv <- bam(y ~ s(sind, bs="cr", k=15) +
                  s(id, by=phi_b_1, bs="re") + s(id, by=phi_b_2, bs="re") +
                  s(id_day, by=phi_nu_1, bs="re") + s(id_day, by=phi_nu_2, bs="re"
                data=df_mfpca_long, method="fREML", discrete=TRUE)
```

Multilevel Functional Data: *mgcv*





References I

Di, C., Crainiceanu, C. M., Caffo, B. S., and Punjabi, N. M. (2009).
Multilevel functional principial component analysis.
Annals of Applied Statistics, 3(1):458–488.