CrossMark

Fast covariance estimation for sparse functional data

Luo Xiao[1] · **Cai Li**[1] · **William Checkley**[2] · **Ciprian Crainiceanu**[3]

Abstract Smoothing of noisy sample covariances is an important component in functional data analysis. We propose a novel covariance smoothing method based on penalized splines and associated software. The proposed method is a bivariate spline smoother that is designed for covariance smoothing and can be used for sparse functional or longitudinal data. We propose a fast algorithm for covariance smoothing using leave-one-subject-out cross-validation. Our simulations show that the proposed method compares favorably against several commonly used methods. The method is applied to a study of child growth led by one of coauthors and to a public dataset of longitudinal CD4 counts.

Keywords Bivariate smoothing · FACEs · fPCA

1 Introduction

The covariance function is a crucial ingredient in functional data analysis. Sparse functional or longitudinal data

✉ Luo Xiao
lxiao5@ncsu.edu

1 Department of Statistics, North Carolina State University, Raleigh, NC, USA

2 School of Medicine, Johns Hopkins University, Baltimore, MD, USA

3 Department of Biostatistics, Johns Hopkins University, Baltimore, MD, USA

are ubiquitous in scientific studies, while functional principal component analysis has become one of the first-line approaches to analyzing this type of data; see, e.g., Besse and Ramsay (1986), Ramsay and Dalzell (1991), Kneip (1994), Besse et al. (1997), Staniswalis and Lee (1998), Yao et al. (2003, 2005).

Given a sample of functions observed at a finite number of locations and, often, with sizable measurement error, there are usually three approaches for obtaining smooth functional principal components: (1) smooth the functional principal components of the sample covariance function; (2) smooth each curve and diagonalize the resulting sample covariance of the smoothed curves; and (3) smooth the sample covariance function and then diagonalize it.

The sample covariance function is typically noisy and difficult to interpret. Therefore, bivariate smoothing is usually employed. Local linear smoothers (Fan and Gijbels 1996), tensor-product bivariate P-splines (Eilers and Marx 2003) and thin plate regression splines (Wood 2003) are among the popular methods for smoothing the sample covariance function. For example, the *fpca.sc* function in the R package *refund* (Huang et al. 2015) uses the tensor-product bivariate P-splines. However, there are two known problems with these smoothers: (1) they are general-purpose smoothers that are not designed specifically for covariance operators; and (2) they ignore that the subject, instead of the observation, is the independent sampling unit and assume that the empirical covariance surface is the sum between an underlying smooth covariance surface and independent random noise. The FACE smoothing approach proposed by Xiao et al. (2016) was designed specifically to address these weaknesses of off-the-shelf covariance smoothing software. The method is implemented in the function *fpca.face* in the *refund* R package (Huang et al. 2015) and has proven to be reliable and fast in a range of applications. However, FACE was developed for

high-dimensional dense functional data and the extension to sparse data is far from obvious. One approach that attempts to solve these problems was proposed by Yao et al. (2003). In their paper, they used leave-one-subject-out cross-validation to choose the bandwidth for local polynomial smoothing methods. This approach is theoretically sound, but computationally expensive. This may be the reason why the practice is to either try multiple bandwidths and visually inspect the results or completely ignore within-subject correlations.

Several alternative methods for covariance smoothing of sparse functional data also exist in the literature: James et al. (2000) used reduced rank spline mixed effects models, Cai and Yuan (2012) considered nonparametric covariance function under the reproducing kernel Hilbert space framework, and Peng and Paul (2009) proposed a geometric approach under the framework of marginal maximum likelihood estimation.

Our paper has two aims. First, we propose a new automatic bivariate smoother that is specifically designed for covariance function estimation and can be used for sparse functional data. Second, we propose a fast algorithm for selecting the smoothing parameter of the bivariate smoother using leave-one-subject-out cross-validation. The code for the proposed method is publicly available in the *face* R package (Xiao et al. 2017).

2 Model

Suppose that the observed data take the form $\{(y_{ij}, t_{ij}), j = 1, \ldots, m_i, i = 1, \ldots, n\}$, where t_{ij} is in the unit interval $[0, 1]$, n is the number of subjects, and m_i is the number of observations for subject i. The model is

$$y_{ij} = f(t_{ij}) + u_i(t_{ij}) + \varepsilon_{ij}, \tag{1}$$

where f is a smooth mean function, $u_i(t)$ is generated from a zero-mean Gaussian process with covariance operator $C(s, t) = \text{cov}\{u_i(s), u_i(t)\}$, and ε_{ij} is white noise following a normal distribution $\mathcal{N}(0, \sigma_\varepsilon^2)$. We assume that the random terms are independent across subjects and from each other. For longitudinal data, m_i's are usually much smaller than n.

We are interested in estimating the covariance function $C(s, t)$. A standard procedure employed for obtaining a smooth estimate of $C(s, t)$ consists of two steps. In the first step, an empirical estimate of the covariance function is constructed. Let $r_{ij} = y_{ij} - f(t_{ij})$ be the residuals and $C_{ij_1 j_2} = r_{ij_1} r_{ij_2}$ be the auxiliary variables. Because $\mathbb{E}(C_{ij_1 j_2}) = C(t_{ij_1}, t_{ij_2})$ if $j_1 \neq j_2$, $\{C_{ij_1 j_2} : 1 \leq j_1 \neq j_2 \leq m_i, i = 1, \ldots, n\}$ is a collection of unbiased empirical estimates of the covariance function. In the second step, the empirical estimates are smoothed using a bivariate smoother. Smoothing is required because the empirical estimates are usually noisy and scattered in time. Standard

bivariate smoothers are local linear smoothers (Fan and Gijbels 1996), tensor-product bivariate *P*-splines (Eilers and Marx 2003) and thin plate regression splines (Wood 2003). In the following section, we propose a statistically efficient, computationally fast and automatic smoothing procedure that serves as an alternative to these approaches.

To carry out the above steps, we assume a mean function estimator $\hat{f}$ exists. Then, we let $\hat{r}_{ij} = y_{ij} - \hat{f}(t_{ij})$ and $\widehat{C}_{ij_1 j_2} = \hat{r}_{ij_1} \hat{r}_{ij_2}$. Note that we use the hat notation on variables when f is substituted by $\hat{f}$ and when we define a variable with a hat notation, the same variable without a hat notation is similarly defined using the true f. In our software, we estimate f using a *P*-spline smoother (Eilers and Marx 1996) with the smoothing parameter selected by leave-one-subject-out cross-validation. See Section S.1 of the online supplement for details.

3 Method

We model the covariance function $C(s, t)$ as a tensor-product splines $H(s, t) = \sum_{1 \leq \kappa \leq c, 1 \leq \ell \leq c} \theta_{\kappa \ell} B_\kappa(s) B_\ell(t)$, where $\Theta = (\theta_{\kappa \ell})_{1 \leq \kappa \leq c, 1 \leq \ell \leq c}$ is a coefficient matrix, $\{B_1(\cdot), \ldots, B_c(\cdot)\}$ is the collection of B-spline basis functions in the unit interval, and c is the number of interior knots plus the order (degree plus 1) of the B-splines. Note that the locations and number of knots as well as the polynomial degrees of splines determine the forms of the B-spline basis functions (de Boor 1978). We use equally spaced knots and enforce the following constraint on Θ:

$$\theta_{\kappa \ell} = \theta_{\ell \kappa}, \quad 1 \leq \kappa, \quad \ell \leq c.$$

With this constraint, $H(s, t)$ is always symmetric in s and t, a desired property for estimates of covariance functions.

Unlike the other approaches covariance function estimation methods described before, our method applies a joint estimation of covariance function and error variance and incorporates the correlation structure of the auxiliary variables $\{\widehat{C}_{ij_1 j_2} : 1 \leq j_1 \leq j_2 \leq m_i, i = 1, \ldots, n\}$ in a two-step procedure to boost statistical efficiency. Because we use a relatively large number of knots, estimating Θ by least squares or weighted least squares tends to overfit. Thus, we estimate Θ by minimizing a penalized weighted least squares. Let $n_i = m_i(m_i + 1)/2$, $\widehat{\mathbf{C}}_{ij} = \{\widehat{C}_{ijj}, \widehat{C}_{ij(j+1)}, \ldots, \widehat{C}_{ijm_i}\}^T \in \mathbb{R}^{m_i - j + 1}$, $\mathbf{H}_{ij} = \{H(t_{ij}, t_{ij}), H(t_{ij}, t_{i(j+1)}), \ldots, H(t_{ij}, t_{im_i})\}^T \in \mathbb{R}^{m_i - j + 1}$, and $\boldsymbol{\delta}_{ij} = (1, \mathbf{0}_{m_i - j}^T)^T \in \mathbb{R}^{m_i - j + 1}$ for $1 \leq j \leq m_i$. Then, let $\widehat{\mathbf{C}}_i = (\widehat{\mathbf{C}}_{i1}^T, \widehat{\mathbf{C}}_{i2}^T, \ldots, \widehat{\mathbf{C}}_{im_i}^T)^T \in \mathbb{R}^{n_i}$ be the vector of all auxiliary variables $\widehat{C}_{ij_1 j_2}$ for subject i with $j_1 \leq j_2$. Here, $\widehat{\mathbf{C}}_i$ contains the nugget terms $\widehat{C}_{ijj}$ and note that $\mathbb{E}(C_{ijj}) = r(t_{ij}, t_{ij}) + \sigma_\varepsilon^2$. Similarly, we let $\mathbf{H}_i = (\mathbf{H}_{i1}^T, \mathbf{H}_{i2}^T, \ldots, \mathbf{H}_{im_i}^T)^T \in \mathbb{R}^{n_i}$, and $\boldsymbol{\delta}_i =$

$(\boldsymbol{\delta}_{i1}^T, \boldsymbol{\delta}_{i2}^T, \ldots, \boldsymbol{\delta}_{im_i}^T)^T \in \mathbb{R}^{n_i}$. Also let $\mathbf{W}_i \in \mathbb{R}^{n_i \times n_i}$ be a weight matrix for capturing the correlation of $\widehat{\mathbf{C}}_i$ and will be specified later. The weighted least squares is WLS $= \sum_{i=1}^{n} (\boldsymbol{H}_i + \boldsymbol{\delta}_i \sigma_\varepsilon^2 - \widehat{\mathbf{C}}_i)^T \mathbf{W}_i (\boldsymbol{H}_i + \boldsymbol{\delta}_i \sigma_\varepsilon^2 - \widehat{\mathbf{C}}_i)$. Let $\|\cdot\|_F$ denote the Frobenius norm and let $\mathbf{D} \in \mathbb{R}^{c \times (c-2)}$ be a second-order differencing matrix (Eilers and Marx 1996). Then, we estimate $\boldsymbol{\Theta}$ and σ_ε^2 by

$$(\widehat{\boldsymbol{\Theta}}, \hat{\sigma}_\varepsilon^2) = \arg \min_{\boldsymbol{\Theta}: \boldsymbol{\Theta} = \boldsymbol{\Theta}^T, \sigma_\varepsilon^2} \left\{ \text{WLS} + \lambda \|\boldsymbol{\Theta} \mathbf{D}\|_F^2 \right\}, \qquad (2)$$

where λ is a smoothing parameter that balances model fit and smoothness of the estimate.

The penalty term $\|\boldsymbol{\Theta} \mathbf{D}\|_F^2$ is essentially equivalent to the penalty $\iint_{s,t} \left\{ \frac{\partial^2 H}{\partial s^2}(s, t) \right\}^2 \mathrm{d}s\mathrm{d}t$ and can be interpreted as the row penalty in bivariate P-splines (Eilers and Marx 2003). Note that when $\boldsymbol{\Theta}$ is symmetric, as in our case, the row and column penalties in bivariate P-splines become the same. Therefore, our proposed method can be regarded as a special case of bivariate P-splines that is designed specifically for covariance function estimation. Another note is that when the smoothing parameter goes to infinity, the penalty term forces $H(s, t)$ to become linear in both the s and the t directions. Finally, if $\hat{\theta}_{\kappa\ell}$ denotes the (κ, ℓ)th element of $\widehat{\boldsymbol{\Theta}}$, then our estimate of the covariance function $C(s, t)$ is given by $\widetilde{C}(s, t) = \sum_{1 \le \kappa \le c, 1 \le \ell \le c} \hat{\theta}_{\kappa\ell} B_\kappa(s) B_\ell(t)$.

3.1 Estimation

Let $\mathbf{b}(t) = \{B_1(t), \ldots, B_c(t)\}^T$ be a vector. Let vec$(\cdot)$ be an operator that stacks the columns of a matrix into a vector and denote $\otimes$ the Kronecker product operator. Then $H(s, t) = \{\mathbf{b}(t) \otimes \mathbf{b}(s)\}^T \text{vec} \boldsymbol{\Theta}$. Let $\boldsymbol{\theta} = \text{vech} \boldsymbol{\Theta}$, where vech$(\cdot)$ is an operator that stacks the columns of the lower triangle of a matrix into a vector, and let $\mathbf{G}_c$ be the duplication matrix (Seber 2007, p. 246) such that $\boldsymbol{\Theta} = \mathbf{G}_c \boldsymbol{\theta}$. It follows that $H(s, t) = \{\mathbf{b}(t) \otimes \mathbf{b}(s)\}^T \mathbf{G}_c \boldsymbol{\theta}$.

Let $\mathbf{B}_{ij} = [\mathbf{b}(t_{ij}), \ldots, \mathbf{b}(t_{im_i})] \otimes \mathbf{b}(t_{ij})$, $\mathbf{B}_i = [\mathbf{B}_{i1}^T, \ldots, \mathbf{B}_{im_i}^T]^T$ and $\mathbf{B} = [\mathbf{B}_1^T, \ldots, \mathbf{B}_n^T]^T$. Also let $\mathbf{X}_i = [\mathbf{B}_i \mathbf{G}_c, \boldsymbol{\delta}_i]$ and $\mathbf{X} = [\mathbf{X}_1^T, \ldots, \mathbf{X}_n^T]^T$. $\boldsymbol{\alpha} = (\boldsymbol{\theta}^T, \sigma_\varepsilon^2)^T$. Finally let $\widehat{\mathbf{C}} = (\widehat{\mathbf{C}}_1^T, \ldots, \widehat{\mathbf{C}}_n^T)^T$, $\boldsymbol{\delta} = (\boldsymbol{\delta}_1^T, \cdots, \boldsymbol{\delta}_n^T)^T$ and $\mathbf{W} = \text{blockdiag}(\mathbf{W}_1, \cdots, \mathbf{W}_n)$. Note that $\mathbf{X}$ can also be written as $[\mathbf{B}\mathbf{G}_c, \boldsymbol{\delta}]$. Then,

$$\mathbb{E}(\widehat{\mathbf{C}}_i) = \boldsymbol{H}_i + \boldsymbol{\delta}_i \sigma_\varepsilon^2 = [\mathbf{B}_i \mathbf{G}_c, \boldsymbol{\delta}_i] (\boldsymbol{\theta}^T, \delta_\varepsilon)^T = \mathbf{X}_i \boldsymbol{\alpha},$$

and

$$\text{WLS} = (\widehat{\mathbf{C}} - \mathbf{X}\boldsymbol{\alpha})^T \mathbf{W} (\widehat{\mathbf{C}} - \mathbf{X}\boldsymbol{\alpha}). \qquad (3)$$

Next let tr$(\cdot)$ be the trace operator such that for a square matrix $\mathbf{A}$, tr$(\mathbf{A})$ is the sum of the diagonals of $\mathbf{A}$. We can derive that (Seber 2007, p. 241)

$$\|\boldsymbol{\Theta} \mathbf{D}\|_F^2 = \text{tr}(\boldsymbol{\Theta} \mathbf{D} \mathbf{D}^T \boldsymbol{\Theta}^T),$$
$$= (\text{vec} \, \boldsymbol{\Theta})^T (\mathbf{I}_c \otimes \mathbf{D} \mathbf{D}^T) \text{vec} \, \boldsymbol{\Theta}.$$

Because $\boldsymbol{\Theta} = \mathbf{G}_c \boldsymbol{\theta}$, we obtain that

$$\|\boldsymbol{\Theta} \mathbf{D}\|_F^2 = \boldsymbol{\theta}^T \mathbf{G}_c^T (\mathbf{I}_c \otimes \mathbf{D} \mathbf{D}^T) \mathbf{G}_c^T \boldsymbol{\theta}$$
$$= (\boldsymbol{\theta}^T \, \sigma_\varepsilon^2) \begin{pmatrix} \mathbf{P} & \mathbf{0} \\ \mathbf{0} & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\theta} \\ \sigma_\varepsilon^2 \end{pmatrix}$$
$$= \boldsymbol{\alpha}^T \mathbf{Q} \boldsymbol{\alpha}, \qquad (4)$$

where $\mathbf{P} = \mathbf{G}_c^T (\mathbf{I}_c \otimes \mathbf{D} \mathbf{D}^T) \mathbf{G}_c^T$ and $\mathbf{Q}$ is the block matrix containing $\mathbf{P}$ and zeros.

By (3) and (4), the objective function in (2) can be rewritten as

$$\hat{\boldsymbol{\alpha}} = \arg \min_{\boldsymbol{\alpha}} (\widehat{\mathbf{C}} - \mathbf{X}\boldsymbol{\alpha})^T \mathbf{W} (\widehat{\mathbf{C}} - \mathbf{X}\boldsymbol{\alpha}) + \lambda \boldsymbol{\alpha}^T \mathbf{Q} \boldsymbol{\alpha}. \qquad (5)$$

Now we obtain an explicit form of $\hat{\boldsymbol{\alpha}}$

$$\hat{\boldsymbol{\alpha}} = \begin{pmatrix} \hat{\boldsymbol{\theta}} \\ \hat{\sigma}_\varepsilon^2 \end{pmatrix} = (\mathbf{X}^T \mathbf{W} \mathbf{X} + \lambda \mathbf{Q})^{-1} (\mathbf{X}^T \mathbf{W} \widehat{\mathbf{C}}). \qquad (6)$$

We need to specify the weight matrices $\mathbf{W}_i$'s. One sensible choice for $\mathbf{W}_i$ is the inverse of cov$(\mathbf{C}_i)$, where $\mathbf{C}_i$ is defined similar to $\widehat{\mathbf{C}}_i$, except that the true mean function f is used. However, cov$(\mathbf{C}_i)$ may not be invertible or may be close to being singular. Thus, we specify $\mathbf{W}_i$ as

$$\mathbf{W}_i^{-1} = (1 - \beta)\text{cov}(\mathbf{C}_i) + \beta \text{diag}\{\text{diag}\{\text{cov}(\mathbf{C}_i)\}\},$$
$$1 \le i \le n,$$

for some constant $0 < \beta < 1$. The above specification ensures that $\mathbf{W}_i$ exists and is stable. We will use $\beta = 0.05$, which works well in practice.

We now derive cov$(\mathbf{C}_i)$ in terms of C and σ_ε^2. First note that $\mathbb{E}(r_{ij_1} r_{ij_2}) = \text{cov}(r_{ij_1}, r_{ij_2}) = C(t_{ij_1}, t_{ij_2}) + \delta_{j_1 j_2} \sigma_\varepsilon^2$, where $\delta_{j_1 j_2} = 1$ if $j_1 = j_2$ and 0 otherwise.

Proposition 1 *Define* $\mathbf{M}_{ijk} = \{C(t_{ij}, t_{ik}), \delta_{jk}\sigma_\varepsilon^2\}^T \in \mathbb{R}^2$. *Then,*

$$cov(C_{ij_1 j_2}, C_{ij_3 j_4}) = \mathbf{1}^T (\mathbf{M}_{ij_1 j_3} \otimes \mathbf{M}_{ij_2 j_4} + \mathbf{M}_{ij_1 j_4} \otimes \mathbf{M}_{ij_2 j_3}).$$

The proof of Proposition 1 is provided in Section S.2 of the online supplement. Now we see that $\mathbf{W}_i$ also depends on $(C, \sigma_\varepsilon^2)$. Hence, we employ a two-stage estimation. We first estimate $(C, \sigma_\varepsilon^2)$ by using penalized ordinary least squares, i.e., $\mathbf{W}_i = \mathbf{I}$ for all i. Then, we obtain the plug-in estimate of $\mathbf{W}_i$ and estimate $(C, \sigma_\varepsilon^2)$ using penalized weighted least squares. The algorithm for the two-stage estimation is summarized as Algorithm 1.

Algorithm 1: Estimation algorithm

Input: data, specification of settings of univariate marginal basis functions and the smoothing parameter λ

Output: estimate of C and σ_ε^2

1 Initialize $\widehat{\mathbf{C}}$, $\mathbf{X}$ and $\mathbf{Q}$;
2 $\hat{\boldsymbol{\alpha}}^{(0)} \leftarrow \arg\min_{\boldsymbol{\alpha}} (\widehat{\mathbf{C}} - \mathbf{X}\boldsymbol{\alpha})^T (\widehat{\mathbf{C}} - \mathbf{X}\boldsymbol{\alpha}) + \lambda \boldsymbol{\alpha}^T \mathbf{Q}\boldsymbol{\alpha}$;
3 $\widehat{\mathbf{W}} \leftarrow \mathbf{W}\{\hat{\boldsymbol{\alpha}}^{(0)}\}$;
4 $\hat{\boldsymbol{\alpha}} \leftarrow \arg\min_{\boldsymbol{\alpha}} (\widehat{\mathbf{C}} - \mathbf{X}\boldsymbol{\alpha})^T \widehat{\mathbf{W}} (\widehat{\mathbf{C}} - \mathbf{X}\boldsymbol{\alpha}) + \lambda \boldsymbol{\alpha}^T \mathbf{Q}\boldsymbol{\alpha}$;

3.2 Selection of the smoothing parameter

For selecting the smoothing parameter, we use leave-one-subject-out cross-validation, a popular approach for correlated data; see, for example, Yao et al. (2003), Reiss et al. (2010) and Xiao et al. (2015). Compared to the leave-one-observation-out cross-validation, which ignores the correlation, leave-one-subject-out cross-validation was reported to be more robust against overfit. However, such an approach is usually computationally expensive. In this section, we derive a fast algorithm for approximating the leave-one-subject-out cross-validation.

Let $\widetilde{\mathbf{C}}_i^{[i]}$ be the prediction of $\widehat{\mathbf{C}}_i$ by applying the proposed method to the data without the data from the ith subject, and then the cross-validated error is

$$\text{iCV} = \sum_{i=1}^{n} \|\widetilde{\mathbf{C}}_i^{[i]} - \widehat{\mathbf{C}}_i\|^2. \tag{7}$$

There is a simple formula for iCV. First, we let $\mathbf{S} = \mathbf{X}(\mathbf{X}^T\mathbf{W}\mathbf{X} + \lambda\mathbf{Q})^{-1}\mathbf{X}^T\mathbf{W}$, which is the smoother matrix for the proposed method. $\mathbf{S}$ can be written as $(\mathbf{X}\mathbf{A})[\mathbf{I} + \lambda\text{diag}(\mathbf{s})]^{-1}(\mathbf{X}\mathbf{A})^T\mathbf{W}$ for some square matrix $\mathbf{A}$ and $\mathbf{s}$ is a column vector; see, for example, Xiao et al. (2013). In particular, both $\mathbf{A}$ and $\mathbf{s}$ do not depend on λ.

Let $\mathbf{S}_i = \mathbf{X}_i(\mathbf{X}^T\mathbf{W}\mathbf{X} + \lambda\mathbf{Q})^{-1}\mathbf{X}^T\mathbf{W}$ and $\mathbf{S}_{ii} = \mathbf{X}_i(\mathbf{X}^T\mathbf{W}\mathbf{X} + \lambda\mathbf{Q})^{-1}\mathbf{X}_i^T\mathbf{W}_i$. Then, $\mathbf{S}_i$ is of dimension $n_i \times N$, where $N = \sum_{i=1}^{n} n_i$, and $\mathbf{S}_{ii}$ is of dimension $n_i \times n_i$.

Lemma 1 *The* iCV *in* (7) *can be simplified as*

$$\text{iCV} = \sum_{i=1}^{n} \|(\mathbf{I}_{n_i} - \mathbf{S}_{ii})^{-1}(\mathbf{S}_i\widehat{\mathbf{C}} - \widehat{\mathbf{C}}_i)\|^2.$$

The proof of Lemma 1 is the same as that of Lemma 3.1 in Xu and Huang (2012) and thus is omitted. Similar to Xu and Huang (2012), we further simplify iCV by using the approximation $(\mathbf{I}_{n_i} - \mathbf{S}_{ii}^T)^{-1}(\mathbf{I}_{n_i} - \mathbf{S}_{ii})^{-1} = \mathbf{I}_{n_i} + \mathbf{S}_{ii} + \mathbf{S}_{ii}^T$. This approximation leads to the generalized cross-validation, which we denote as iGCV,

$$\text{iGCV} = \sum_{i=1}^{n} (\mathbf{S}_i\widehat{\mathbf{C}} - \widehat{\mathbf{C}}_i)^T (\mathbf{I}_{n_i} + \mathbf{S}_{ii} + \mathbf{S}_{ii}^T)(\mathbf{S}_i\widehat{\mathbf{C}} - \widehat{\mathbf{C}}_i)$$

$$= \|\widehat{\mathbf{C}} - \mathbf{S}\widehat{\mathbf{C}}\|^2 + 2\sum_{i=1}^{n} (\mathbf{S}_i\widehat{\mathbf{C}} - \widehat{\mathbf{C}}_i)^T$$

$$\mathbf{S}_{ii}(\mathbf{S}_i\widehat{\mathbf{C}} - \widehat{\mathbf{C}}_i). \tag{8}$$

While iGCV in (8) is much easier to compute than iCV in (7), the formula in (8) is still computationally expensive as the smoother matrix $\mathbf{S}$ is of dimension $N \times N$, where $N = 2,000$ if $n = 100$ and $m_i = m = 5$ for all i. Thus, we further simplify iGCV.

Let $\mathbf{F}_i = \mathbf{X}_i\mathbf{A}$, $\mathbf{F} = \mathbf{X}\mathbf{A}$ and $\widetilde{\mathbf{F}} = \mathbf{F}^T\mathbf{W}$. Define $\boldsymbol{f}_i = \mathbf{F}_i^T\widehat{\mathbf{C}}_i$, $\boldsymbol{f} = \mathbf{F}^T\widehat{\mathbf{C}}$, $\tilde{\mathbf{f}} = \widetilde{\mathbf{F}}\widehat{\mathbf{C}}$, $\mathbf{J}_i = \mathbf{F}_i^T\mathbf{W}_i\widehat{\mathbf{C}}_i$, $\mathbf{L}_i = \mathbf{F}_i^T\mathbf{F}_i$ and $\widetilde{\mathbf{L}}_i = \mathbf{F}_i^T\mathbf{W}_i\mathbf{F}_i$. To simplify notation we will denote $[\mathbf{I} + \lambda\text{diag}(\mathbf{s})]^{-1}$ as $\widetilde{\mathbf{D}}$, a symmetric matrix, and its diagonal as $\tilde{\mathbf{d}}$. Let $\odot$ be the Hadamard product such that for two matrices of the same dimensions $A = (a_{ij})$ and $B = (b_{ij})$, $A \odot B = (a_{ij}b_{ij})$.

Proposition 2 *The iGCV in* (8) *can be simplified as*

$$iGCV = \|\widehat{\mathbf{C}}\|^2 - 2\tilde{\mathbf{d}}^T(\tilde{\mathbf{f}} \odot \boldsymbol{f}) + (\tilde{\mathbf{f}} \odot \tilde{\mathbf{d}})^T(\mathbf{F}^T\mathbf{F})(\tilde{\mathbf{f}} \odot \tilde{\mathbf{d}})$$

$$+ 2\tilde{\mathbf{d}}^T\boldsymbol{g} - 4\tilde{\mathbf{d}}^T\mathbf{G}\tilde{\mathbf{d}} + 2\tilde{\mathbf{d}}^T$$

$$\left[\sum_{i=1}^{n} \{\mathbf{L}_i(\tilde{\mathbf{f}} \odot \tilde{\mathbf{d}})\} \odot \{\widetilde{\mathbf{L}}_i(\tilde{\mathbf{f}} \odot \tilde{\mathbf{d}})\}\right],$$

where $\boldsymbol{g} = \sum_{i=1}^{n} \mathbf{J}_i \odot \boldsymbol{f}_i$ *and* $\mathbf{G} = \sum_{i=1}^{n} (\mathbf{J}_i\tilde{\mathbf{f}}^T) \odot \mathbf{L}_i$.

The proof of Proposition 2 is provided in Section S.2 of the online supplement.

Algorithm 2: Tuning algorithm

Input: $\mathbf{X}$, $\widehat{\mathbf{C}}$, $\mathbf{Q}$, $\mathbf{W}$, $\boldsymbol{\lambda} = \{\lambda_1, \ldots, \lambda_k\}^T$

Output: λ^*

1 Initialize $\mathbf{s}$, $\tilde{\mathbf{f}}$, $\boldsymbol{f}$, $\mathbf{F}$, $\boldsymbol{g}$, $\mathbf{G}$, $\mathbf{L}_i$, $\widetilde{\mathbf{L}}_i$, $i = 1, \ldots, n$;
2 **foreach** λ *in* $\boldsymbol{\lambda}$ **do**
3    $\tilde{\mathbf{d}} \leftarrow \text{diag}([\mathbf{I} + \lambda\text{diag}(\mathbf{s})]^{-1})$;
4    $I \leftarrow -2\tilde{\mathbf{d}}^T(\tilde{\mathbf{f}} \odot \boldsymbol{f})$;
5    $II \leftarrow (\tilde{\mathbf{f}} \odot \tilde{\mathbf{d}})^T(\mathbf{F}^T\mathbf{F})(\tilde{\mathbf{f}} \odot \tilde{\mathbf{d}})$;
6    $III \leftarrow 2\tilde{\mathbf{d}}^T\boldsymbol{g}$;
7    $IV \leftarrow -4\tilde{\mathbf{d}}^T\mathbf{G}\tilde{\mathbf{d}}$;
8    $V \leftarrow 2\tilde{\mathbf{d}}^T[\sum_{i=1}^{n} \{\mathbf{L}_i(\tilde{\mathbf{f}} \odot \tilde{\mathbf{d}})\} \odot \{\widetilde{\mathbf{L}}_i(\tilde{\mathbf{f}} \odot \tilde{\mathbf{d}})\}]$;
9    $\text{iGCV} \leftarrow I + II + III + IV + V$;
10 **end**
11 $\lambda^* \leftarrow \arg\min_\lambda \text{iGCV}$;

While the formula in Proposition 2 looks complex, it can be efficiently computed. Indeed, only the term $\tilde{\mathbf{d}}$ depends on the smoothing parameter λ and it can be easily computed; all other terms including $\boldsymbol{g}$ and $\mathbf{G}$ can be pre-calculated just once. Suppose the number of observations per subject is $m_i = m$

for all i. Let $K = c(c + 1)/2 + 1$ and $M = m(m + 1)/2$. Note that K is the number of unknown coefficients and M is the number of raw covariances from each subject. Then, the pre-calculation of terms in the iGCV formula requires $O(nMK^2 + nM^2K + K^3 + M^3)$ computation time and each calculation of iGCV requires $O(nK^2)$ computation time. To see the efficiency of the simplified formula in Proposition 2, we note that a brute force evaluation of iCV in Lemma 1 requires computation time of the order $O(nM^3 + nK^3 + n^2M^2K)$, quadratic in the number of subjects n.

When the number of observations per subject m is small, i.e., $m < c$, the number of univariate basis functions, the iGCV computation time increases linearly with respect to m; when m is relatively large, i.e., $m > c$ but $m = o(n)$, the iGCV computation time increases quadratically with respect to m. Therefore, the iGCV formula is most efficient with a small m, i.e., sparse data. As for the case that m is very large and the proposed method becomes very slow, the method in Xiao et al. (2016) might be preferred.

4 Curve prediction

In this section, we consider the prediction of $X_i(t) = f(t) + u_i(t)$, the ith subject curve. We assume that $X_i(t)$ is generated from a Gaussian process. Suppose we would like to predict $X_i(t)$ at $\{s_{i1}, \ldots, s_{im}\}$ for $m \geq 1$. Let $\mathbf{y}_i = (\mathbf{y}_{i1}, \ldots, \mathbf{y}_{im_i})^T$, $\boldsymbol{f}_i^o = \{f(t_{i1}), \ldots, f(t_{im_i})\}^T$, and $\mathbf{x}_i = \{X_i(s_{i1}), \ldots, X_i(s_{im})\}^T$. Let $\mathbf{H}_i^o = [\mathbf{b}(t_{i1}), \ldots, \mathbf{b}(t_{im_i})]^T$ and $\mathbf{H}_i^n = [\mathbf{b}(s_{i1}), \ldots, \mathbf{b}(s_{im})]^T$. It follows that

$$\begin{pmatrix} \mathbf{y}_i \\ \mathbf{x}_i \end{pmatrix} \sim \mathcal{N} \left\{ \begin{pmatrix} \boldsymbol{f}_i^o \\ \boldsymbol{f}_i^n \end{pmatrix}, \begin{pmatrix} \mathbf{H}_i^o \boldsymbol{\Theta} \mathbf{H}_i^{o,T} & \mathbf{H}_i^o \boldsymbol{\Theta} \mathbf{H}_i^{n,T} \\ \mathbf{H}_i^n \boldsymbol{\Theta} \mathbf{H}_i^{o,T} & \mathbf{H}_i^n \boldsymbol{\Theta} \mathbf{H}_i^{n,T} \end{pmatrix} \right.$$
$$\left. + \sigma_\varepsilon^2 \mathbf{I}_{m_i+m} \right\}.$$

We derive that

$$\mathbb{E}(\mathbf{x}_i | \mathbf{y}_i) = \left(\mathbf{H}_i^n \boldsymbol{\Theta} \mathbf{H}_i^{o,T} \right) \mathbf{V}_i^{-1} (\mathbf{y}_i - \boldsymbol{f}_i^o) + \boldsymbol{f}_i^n,$$

where $\mathbf{V}_i = \mathbf{H}_i^o \boldsymbol{\Theta} \mathbf{H}_i^{o,T} + \sigma_\varepsilon^2 \mathbf{I}_{m_i}$, and

$$\mathrm{cov}(\mathbf{x}_i | \mathbf{y}_i) = \mathbf{V}_i^n - \left(\mathbf{H}_i^n \boldsymbol{\Theta} \mathbf{H}_i^{o,T} \right) \mathbf{V}_i^{-1} \left(\mathbf{H}_i^n \boldsymbol{\Theta} \mathbf{H}_i^{o,T} \right)^T,$$

where $\mathbf{V}_i^n = \mathbf{H}_i^n \boldsymbol{\Theta} \mathbf{H}_i^{n,T} + \sigma_\varepsilon^2 \mathbf{I}_m$. Because f, $\boldsymbol{\Theta}$ and σ_ε^2 are unknown, we need to plug in their estimates $\hat{f}$, $\widehat{\boldsymbol{\Theta}}$ and $\hat{\sigma}_\varepsilon^2$, respectively, into the above equalities. Thus, we could predict $\mathbf{x}_i$ by

$$\hat{\mathbf{x}}_i = \{\hat{x}_i(s_{i1}), \ldots, \hat{x}_i(s_{im})\}^T$$
$$= \left(\mathbf{H}_i^n \widehat{\boldsymbol{\Theta}} \mathbf{H}_i^{o,T} \right) \hat{\mathbf{V}}_i^{-1} (\mathbf{y}_i - \hat{\boldsymbol{f}}_i^o) + \hat{\boldsymbol{f}}_i^n,$$

where $\hat{\boldsymbol{f}}_i^o = \{\hat{f}(t_{i1}), \ldots, \hat{f}(t_{im_i})\}^T$, $\hat{\boldsymbol{f}}_i^n = \{\hat{f}(s_{i1}), \ldots, \hat{f}(s_{im})\}^T$, and $\hat{\mathbf{V}}_i = \mathbf{H}_i^o \widehat{\boldsymbol{\Theta}} \mathbf{H}_i^{o,T} + \hat{\sigma}_\varepsilon^2 \mathbf{I}_{m_i}$. Moreover, an approximate covariance matrix for $\hat{\mathbf{x}}_i$ is

$$\widehat{\mathrm{cov}}(\hat{\mathbf{x}}_i | \mathbf{y}_i) = \hat{\mathbf{V}}_i^n - \left(\mathbf{H}_i^n \widehat{\boldsymbol{\Theta}} \mathbf{H}_i^{o,T} \right) \hat{\mathbf{V}}_i^{-1} \left(\mathbf{H}_i^n \widehat{\boldsymbol{\Theta}} \mathbf{H}_i^{o,T} \right)^T,$$

where $\hat{\mathbf{V}}_i^n = \mathbf{H}_i^n \widehat{\boldsymbol{\Theta}} \mathbf{H}_i^{n,T} + \hat{\sigma}_\varepsilon^2 \mathbf{I}_m$.

Note that one may also use the standard Karhunen–Loeve decomposition representation of $X_i(t)$ for prediction; see, e.g., Yao et al. (2005). An advantage of the above formulation is that we avoid the evaluation of the eigenfunctions extracted from the covariance function C; indeed, we just need to compute the B-spline basis functions at the desired time points, which is computationally simple.

5 Simulations

5.1 Simulation setting

We generate data using model (1). The number of observations for each random curve is generated from a uniform distribution on either $\{3, 4, 5, 6, 7\}$ or $\{j : 5 \leq j \leq 15\}$, and then observations are sampled from a uniform distribution in the unit interval. Therefore, on average, each curve has $m = 5$ or $m = 10$ observations. The mean function is $\mu(t) = 5 \sin(2\pi t)$. For the covariance function $C(s, t)$, we consider two cases. For case 1, we let $C_1(s, t) = \sum_{\ell=1}^3 \lambda_\ell \psi_\ell(s) \psi_\ell(t)$, where ψ_ℓ's are eigenfunctions and λ_ℓ's are eigenvalues. Here, $\lambda_\ell = 0.5^{\ell-1}$ for $\ell = 1, 2, 3$ and $\psi_1(t) = \sqrt{2} \sin(2\pi t)$, $\psi_2(t) = \sqrt{2} \cos(4\pi t)$ and $\psi_3(t) = \sqrt{2} \sin(4\pi t)$. For case 2, we consider the Matern covariance function

$$C(d; \phi, \nu) = \frac{1}{2^{\nu-1} \Gamma(\nu)} \left(\frac{\sqrt{2\nu}d}{\phi} \right)^\nu K_\nu \left(\frac{\sqrt{2\nu}d}{\phi} \right)$$

with range $\phi = 0.07$ and order $\nu = 1$. Here, K_ν is the modified Bessel function of order ν. The top two eigenvalues for this covariance function are 0.209 and 0.179, respectively. The noise term ε_{ij}'s are assumed normal with mean zero and variance σ_ε^2. We consider two levels of signal to noise ratio (SNR): 2 and 5. For example, if

$$\sigma_\varepsilon^2 = \frac{1}{2} \int_{s=0}^1 \int_{t=0}^1 C(s, t) \mathrm{d}s \mathrm{d}t,$$

then the signal-to-noise ratio in the data is 2. The number of curves is $n = 100$ or 400 and for each covariance function 200 datasets are drawn. Therefore, we have 16 different model conditions to examine.

5.2 Competing methods and evaluation criterion

We compare the proposed method (denoted by FACEs) with the following methods: (1) The *fpca.sc* method in Goldsmith et al. (2010), which uses tensor-product bivariate *P*-splines (Eilers and Marx 2003) for covariance smoothing and is implemented in the R package *refund*; (2) a variant of *fpca.sc* that uses thin plate regression splines for covariance smoothing, denoted by TPRS, and is coded by the authors; 3) the MLE method in Peng and Paul (2009), implemented in the R package *fpca*; and 4) the local polynomial method in Yao et al. (2003), denoted by *loc*, and is implemented in the MATLAB toolbox *PACE*. The underlying covariance smoothing R function for *fpca.sc* and TPRS is *gam* in the R package *mgcv* (Wood 2013). For FACEs, we use $c = 10$ marginal cubic B-spline bases in each dimension. To evaluate the effect of the weight matrices in the proposed objective function (2), we also report results of FACEs without using weight matrices; we denote the one stage fit by FACEs (1-stage). For *fpca.sc*, we use its default setting, which uses 10 B-spline bases in each dimension and the smoothing parameters are selected by "REML." We also code *fpca.sc* ourselves because the *fpca.sc* function in the *refund* R package incorporates other functionalities and may become very slow. For TPRS, we also use the default setting in *gam*, with the smoothing parameter selected by "REML." For bivariate smoothing, the default TPRS uses 27 nonlinear basis functions, in addition to the linear basis functions. We also consider TPRS with 97 nonlinear basis functions to match the basis dimension used in *fpca.sc* and FACEs. For the method MLE, we specify the range for the number of B-spline bases to be [6, 10] and the range of possible ranks to be [2, 6]. We will not evaluate the method using a reduced rank mixed effects model (James et al. 2000) because it has been shown in Peng and Paul (2009) that the MLE method is more superior.

We evaluate the above methods using four criterions. The first is the integrated squared errors (ISE) for estimating the covariance function. The next two criterions are based on the eigendecomposition of the covariance function: $C(s, t) = \sum_{\ell=1}^{\infty} \lambda_\ell \psi_\ell(s) \psi_\ell(t)$, where $\lambda_1 \geq \lambda_2 \geq \ldots$ are eigenvalues and $\psi_1(t), \psi_2(t), \ldots$ are the associated orthonormal eigenfunctions. The second criterion is the integrated squared errors (ISE) for estimating the top 3 eigenfunctions from the covariance function. Let $\psi(t)$ be the true eigenfunction and $\hat{\psi}(t)$ be an estimate of $\psi(t)$, then the integrated squared error is

$$\min \left[\int_{t=0}^{1} \{\psi(t) - \hat{\psi}(t)\}^2 \mathrm{d}t, \int_{t=0}^{1} \{\psi(t) + \hat{\psi}(t)\}^2 \mathrm{d}t \right].$$

It is easy to show that the range of integrated squared error for eigenfunction estimation is [0, 2]. Note that for the method MLE, if rank 2 is selected, then only two eigenfunctions can

be extracted. In this case, to evaluate accuracy of estimating the third eigenfunction, we will let ISE be 1 for a fair comparison. The third criterion is the squared errors (SE) for estimating the top 3 eigenvalues. The last criterion is the methods' computation speed.

5.3 Simulation results

The detailed simulation results are presented in Section S.3 of the online supplement. Here, we provide summaries of the results along with some illustrations. In terms of estimating the covariance function, for most model conditions, FACEs gives the smallest medians of integrated squared errors and has the smallest inter-quarter ranges (IQRs). MLE is the 2nd best for case 1, while *loc* is the 2nd best for case 2. See Figs. 1 and 2 for illustrations under some model conditions.

In terms of estimating the eigenfunctions, FACEs tends to outperform other approaches in most scenarios, while for the remaining scenarios, its performance is still comparable with the best one. MLE performs well for case 1 but relatively poorly for case 2, while the opposite is true for *loc*. TPRS and *fpca.sc* perform quite poorly for estimating the 2nd and 3rd eigenfunctions in both case 1 and case 2. Figure 3 illustrates the superiority of FACEs for estimating eigenfunctions when $n = 100, m = 5$.

As for estimation of eigenvalues, we have the following findings: (1) FACEs performs the best for estimating the first eigenvalue in case 1; (2) *loc* performs the best for estimating the first eigenvalue in case 2; (3) MLE performs overall the best for estimating 2nd and 3rd eigenvalues in both cases, while the performance of FACEs is very close and can be better than MLE under some model scenarios; (4) TPRS, *fpca.sc* and *loc* perform quite poorly for estimating the 2nd and 3rd eigenvalues in most scenarios. We conclude that FACEs shows overall very competitive performance and never deviates much from the best performance. Figure 4 illustrates the patterns of eigenvalue estimation for $n = 100, m = 5$.

We now compare run times of the various methods; see Fig. 5 for an illustration. When $m = 5$, FACEs takes about four to seven times the computation times of TPRS and *fpca.sc*; but it is much faster than MLE and *loc*, the speedup is about 15 and 35 folds, respectively. When $m = 10$, although FACEs is still slower than TPRS and *fpca.sc*, the computation times are similar; computation times of MLE and *loc* are over 9 and 10 folds of FACEs, respectively. Because TPRS and *fpca.sc* are naive covariance smoothers, their fast speed is offset by their tendency to have inferior performance in terms of estimation of covariance functions, eigenfunctions, and eigenvalues.

Finally, by comparing results of FACEs with its 1-stage counterpart (see the online supplement), we see that taking into account of the correlations in the raw covariances boosts the estimation accuracies of FACEs a lot. The 1-stage FACEs
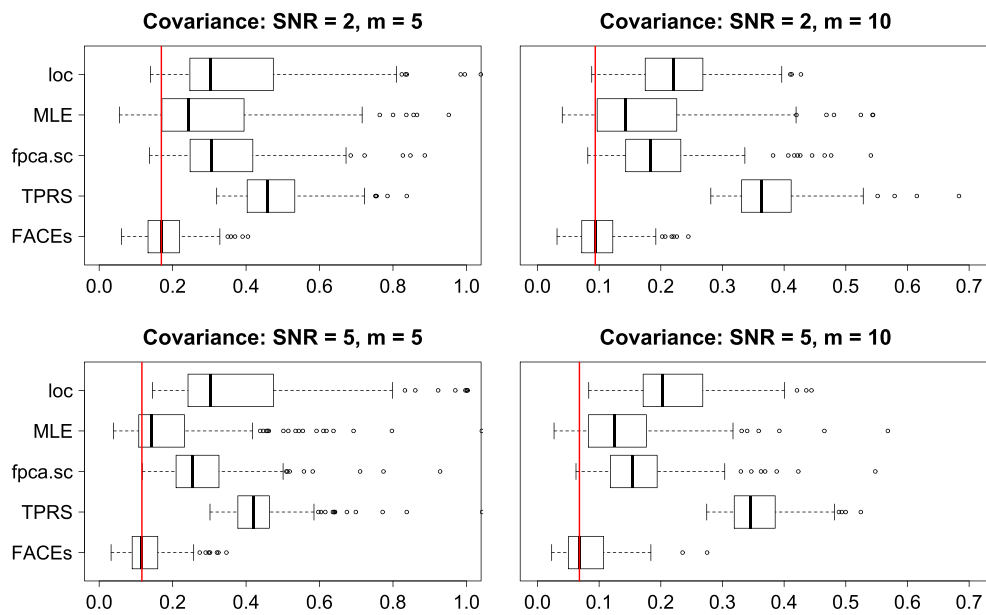
Fig. 1 Boxplots of ISEs of five estimators for estimating the covariance functions of case 1, $n = 100$
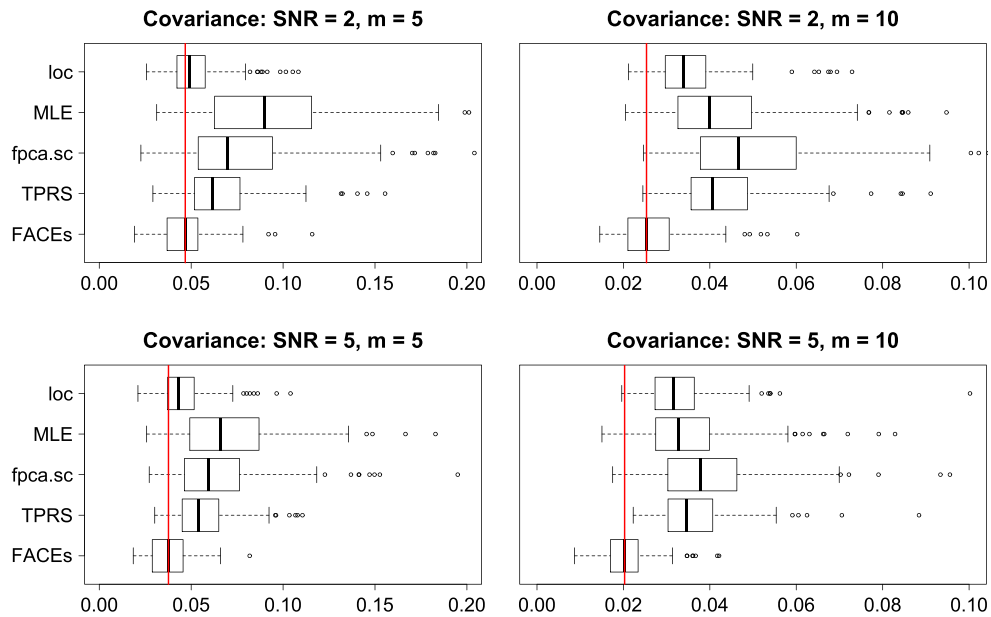

Fig. 2 Boxplots of ISEs of five estimators for estimating the covariance functions of case 2, $n = 100$

is of course faster. It is interesting to note that the 1-stage FACEs is actually also very competitive against other methods.

To summarize, FACEs is a relatively fast method coupled with competing performance against the methods examined above.

5.4 Additional simulations for curve prediction

We conduct additional simulations to evaluate the performance of the FACEs method for curve prediction. We focus on case 1 and use the same simulation settings in Sect. 5.1 for generating the training data and the testing data. We generate 200 new subjects for testing. The number of observations for the subjects are generated in the same way as the training data.

In addition to the conditional expectation approach outlined in Sect. 4, Cederbaum et al. (2016) proposed a new prediction approach (denoted by FAMM). As functional data have a mixed effects representation conditional on eigenfunctions, the standard prediction procedure for mixed effects models can be used for curve prediction. The FAMM requires estimates of eigenfunctions and is applicable to any covari-
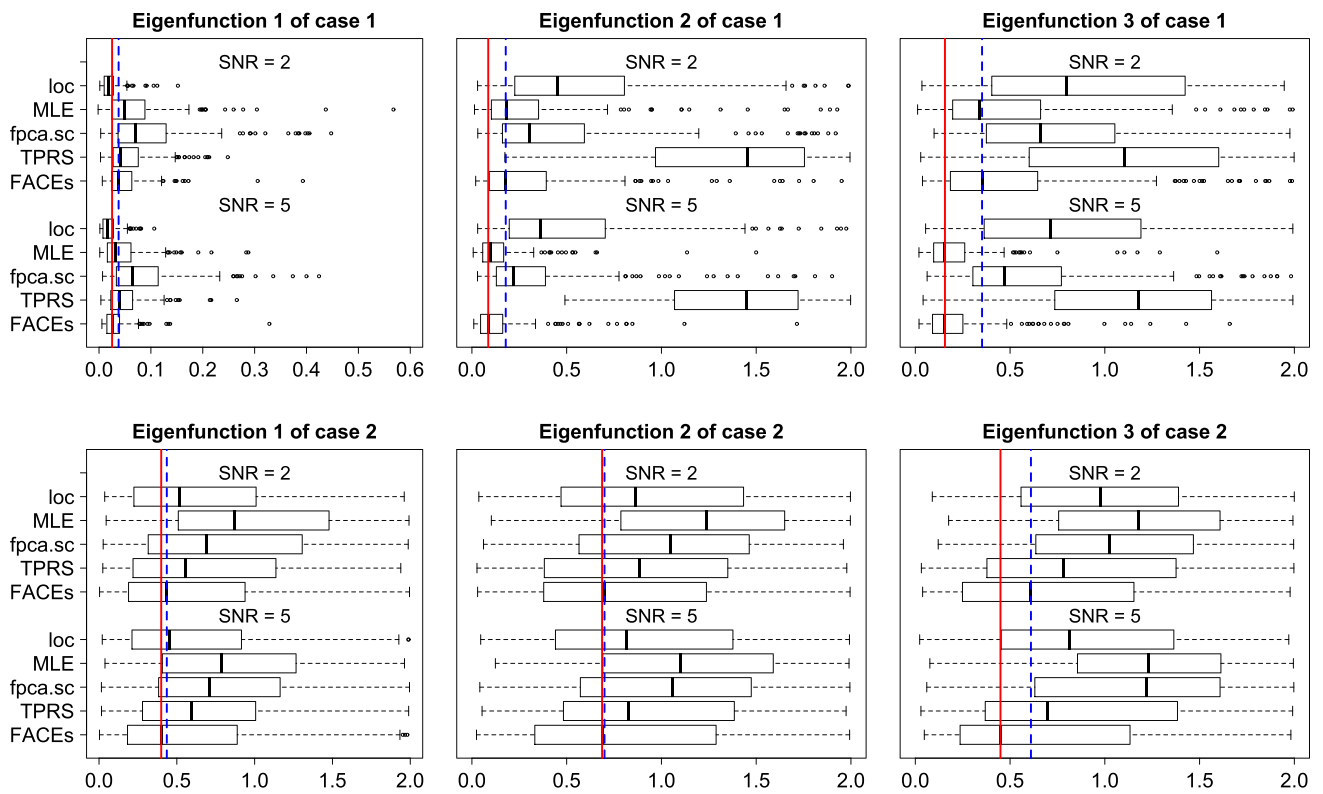
Fig. 3 Boxplots of ISEs of five estimators for estimating the top 3 eigenfunctions when $n = 100, m = 5$. Note that the *straight lines* are the medians of FACEs when SNR $= 5$ and the *dash lines* are the medians of FACEs when SNR $= 2$
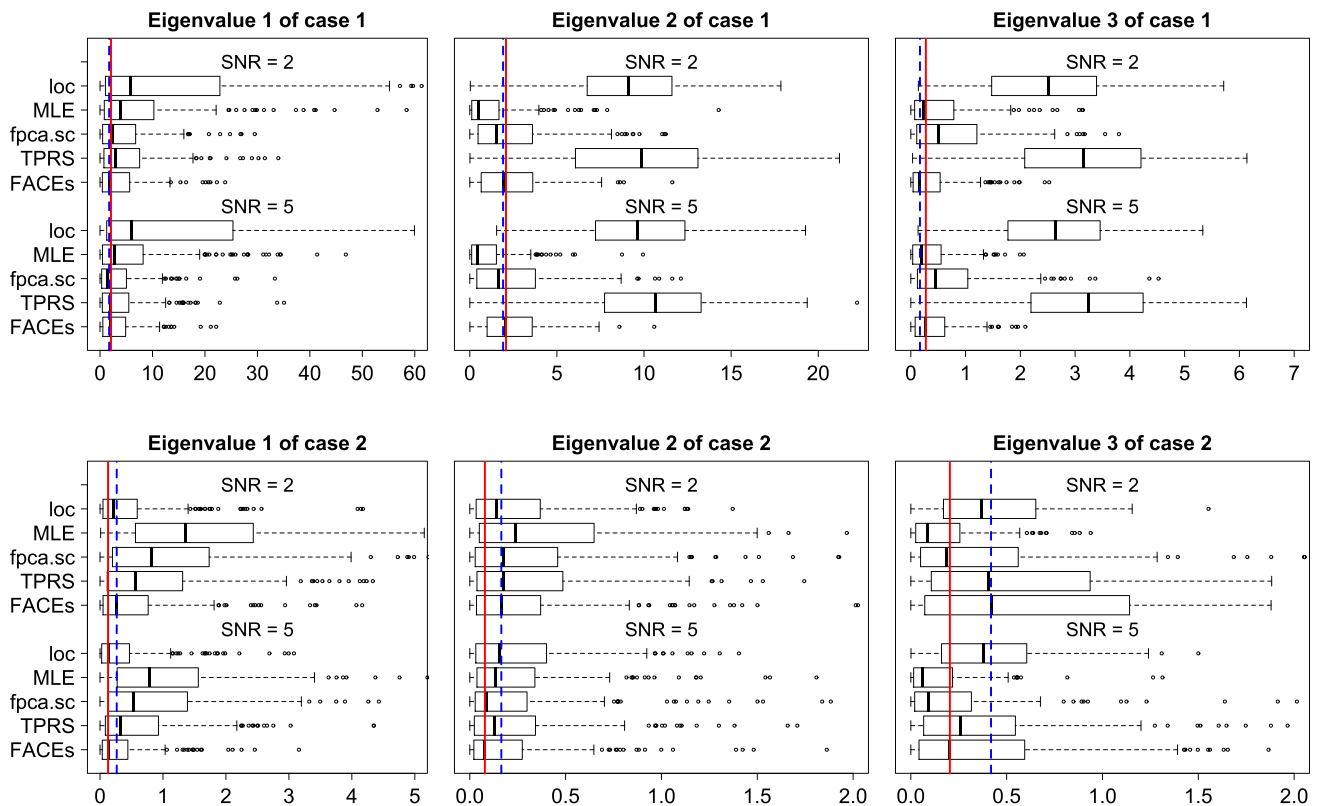

Fig. 4 Boxplots of $100\times$ SEs of five estimators for estimating the eigenvalues when $n = 100, m = 5$. Note that the *straight lines* are the medians of FACEs when SNR $= 5$ and the *dash lines* are the medians of FACEs when SNR $= 2$

Fig. 5 Boxplots of computation times (in seconds) of five estimators for estimating the covariance functions when $n = 400$, SNR = 2. Note that the x-axis is not equally spaced
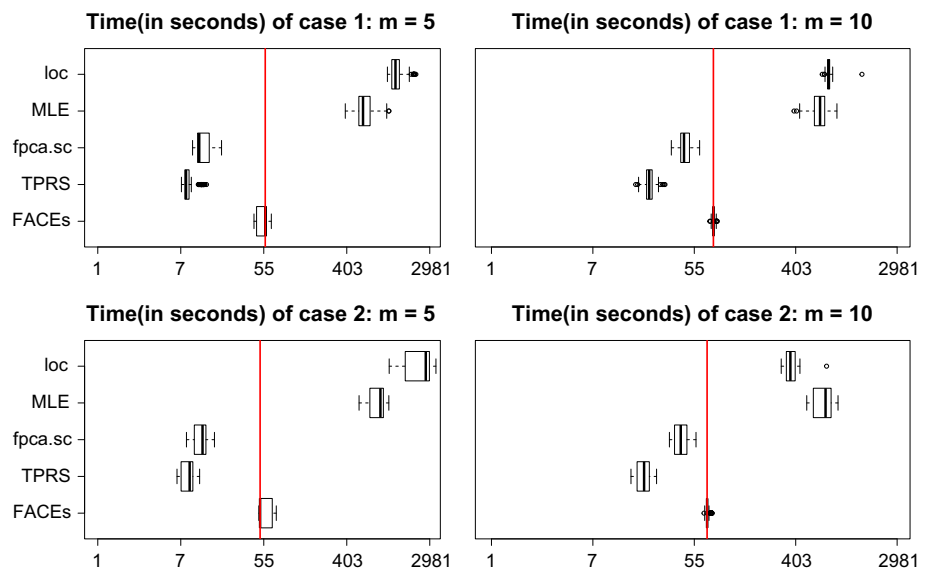
ance smoothing method. Finally, direct estimation of subject-specific curves has also been proposed in the literature (Durban et al. 2005; Chen and Wang 2011; Scheipl et al. 2015).

We will compare the following methods: (1) the conditional expectation method using FACEs; (2) the conditional expectation method using *fpca.sc*; (3) the conditional FAMM method using FACEs; (4) the conditional FAMM method using *fpca.sc*; (5) the conditional expectation method using *loc*; and (6) the spline-based approach in Scheipl et al. (2015) without estimating covariance function, denoted by *pffr*, and is implemented in the R package *refund*. This method uses direct estimation of subject-specific curves. For the conditional FAMM approach, we follow Cederbaum et al. (2016) and fix smoothing parameters at the ratios of the estimated eigenvalues and error variance from covariance function. Fixing smoothing parameters significantly reduces the computation times of the FAMM approach.

We evaluate the above methods using the integrated squared errors, and the results are summarized in Table 1. The results show that either approach (conditional expectation or

conditional FAMM) using FACEs has overall smaller prediction errors than competing approaches. The conditional FAMM approach using FACEs is slightly better than the conditional expectation approach. The results suggest that better estimation of the covariance function leads to more accurate prediction of subject-specific curves.

6 Applications

We illustrate the proposed method on a publicly available dataset. Another application on a child growth dataset is provided in Section S.4 of the online supplement.

CD4 cells are a type of white blood cells that could send signals to the human body to activate the immune response when they detect viruses or bacteria. Thus, the CD4 count is an important biomarker used for assessing the health of HIV-infected persons as HIV viruses attack and destroy the CD4 cells. The dataset analyzed here is from the Multicenter AIDS Cohort Study (MACS) and is available in the *refund* R package (Huang et al. 2015). The observations are CD4 cell

Table 1 Median and IQR (in parenthesis) of ISEs for curve fitting for case 1

n	m	SNR	FACEs	*FAMM*(FACEs)	*fpca.sc*	*FAMM*(fpca.sc)	*loc*	*pffr*
100	5	2	0.714 (0.085)	**0.699** (0.102)	0.790 (0.156)	0.765 (0.147)	0.826 (0.135)	1.178 (0.092)
400	5	2	**0.592** (0.058)	0.596 (0.058)	0.625 (0.077)	0.639 (0.076)	0.735 (0.082)	1.181 (0.093)
100	10	2	0.369 (0.047)	**0.355** (0.044)	0.420 (0.066)	0.405 (0.069)	0.456 (0.076)	0.880 (0.060)
400	10	2	0.323 (0.027)	**0.317** (0.031)	0.330 (0.036)	0.336 (0.035)	0.406 (0.042)	0.872 (0.065)
100	5	5	0.497 (0.074)	**0.476** (0.082)	0.617 (0.171)	0.585 (0.147)	0.636 (0.106)	1.080 (0.109)
400	5	5	0.375 (0.042)	**0.372** (0.042)	0.416 (0.060)	0.419 (0.055)	0.523 (0.066)	1.050 (0.101)
100	10	5	0.218 (0.044)	**0.202** (0.040)	0.259 (0.056)	0.246 (0.053)	0.294 (0.058)	0.734 (0.071)
400	10	5	0.164 (0.019)	**0.160** (0.021)	0.182 (0.028)	0.180 (0.026)	0.243 (0.034)	0.740 (0.066)

The results are based on 200 replications. Numbers in boldface are the smallest of each row
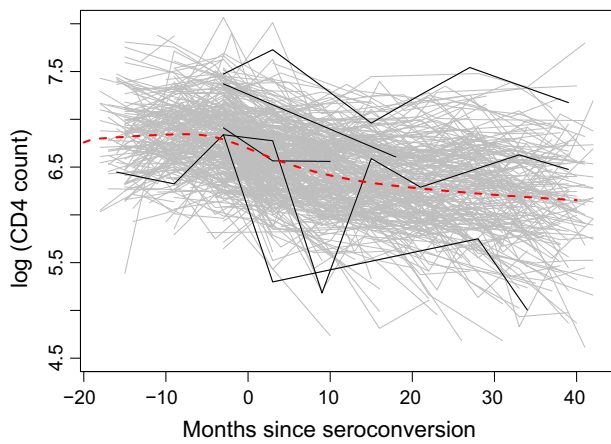
Fig. 6 Observed log (CD4 count) trajectories of 366 HIV-infected males. The estimated population mean is the *black solid line*

counts for 366 infected males in a longitudinal study (Kaslow et al. 1987). With a total of 1888 data points, each subject has between 1 and 11 observations. Statistical analysis based on this or related datasets were done in Diggle et al. (1994), Yao et al. (2005), Peng and Paul (2009) and Goldsmith et al. (2013).

For our analysis, we consider log (CD4 count) since the counts are skewed. We plot the data in Fig. 6 where the *x*-axis is months since seroconversion (i.e., the time at which HIV becomes detectable). The overall trend seems to be decreasing, as can be visually confirmed by the estimated mean function plotted in Fig. 6. The estimated variance and correlation functions are displayed in Fig. 7. It is interesting to see that the minimal value of the estimated variance function occurs at month 0 since seroconversion. Finally, we display

Fig. 7 Estimated variance function (*left panel*) and correlation function (*right panel*) for the log (CD4 count)
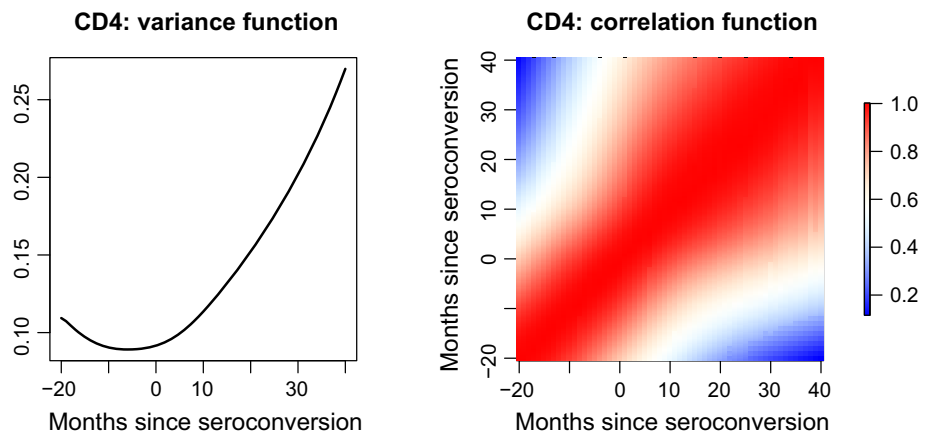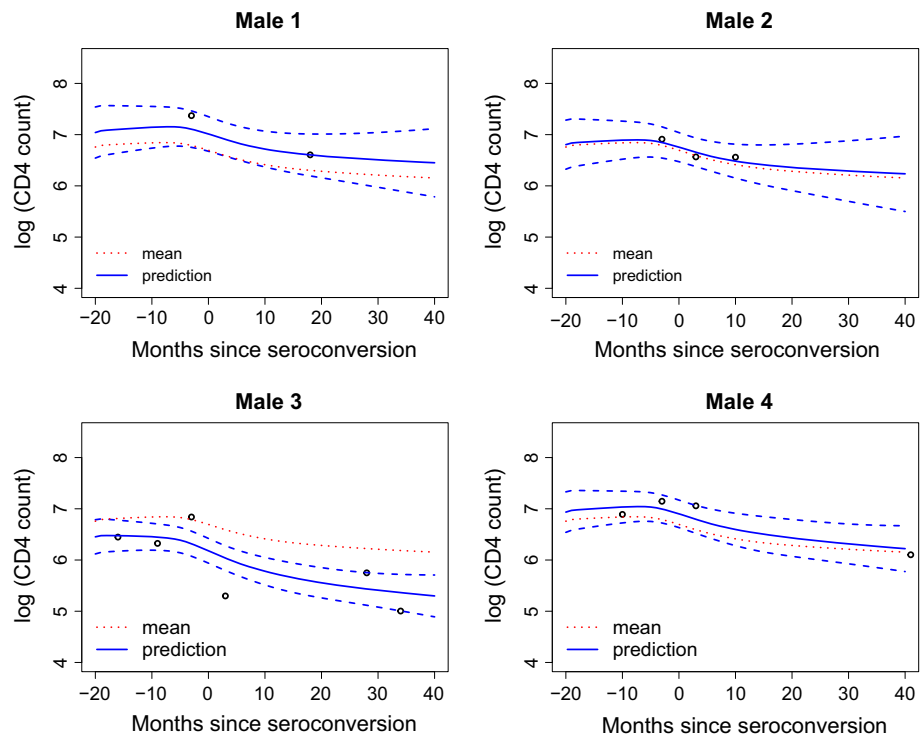

Fig. 8 Predicted subject-specific trajectories of log (CD4 count) and associated 95% confidence bands for 4 males. The estimated population mean is the *dotted line*

in Fig. 8 the predicted trajectory of log (CD4 count) for 4 males and the corresponding pointwise confidence bands.

7 Discussion

Estimating and smoothing covariance operators is an old problem with many proposed solutions. Automatic and fast covariance smoothing is not fully developed, and, in practice, one still does not have a method that is used consistently. The reason why the practical solution to the problem has been quite elusive is the lack of automatic covariance smoothing software. The novelty of our proposal is that it directly tackles this problem from the point of view of practicality. Here, we proposed a method that we are already using extensively in practice and which is becoming increasingly popular among practitioners.

The ingredients of the proposed approach are not all new, but their combination leads to a complete product that can be used in practice. The fundamentally novel contributions that make everything practical are: (1) use a particular type of penalty that respects the covariance matrix format; (2) provide a very fast fitting algorithm for leave-one-subject-out cross-validation; and (3) ensure the scalability of the approach by controlling the overall complexity of the algorithm.

Smoothing parameters are an important component in smoothing and usually selected by either cross-validation or likelihood-based approaches. The latter make use of the mixed model representation of spline-based smoothing (Ruppert et al. 2003) and tend to perform better than cross-validation (Reiss and Todd Ogden 2009; Wood 2011). New optimization techniques have been developed (Rodríguez-Álvarez et al. 2015, 2016; Wood and Fasiolo 2017) for likelihood-based approaches. Likelihood-based approaches seem impractical for smoothing of raw covariances because the entries are products of normal residuals. Moreover, the raw covariances are dependent within subjects, which imposes additional challenge. Developing likelihood-based selection of smoothing parameters for covariance smoothing is of interest but beyond the scope of the paper.

To make methods transparent and reproducible, the method has been made publicly available in the *face* package and will be incorporated in the function *fpca.face* in the *refund* package later. The current *fpca.face* function (Xiao et al. 2016) deals with high-dimensional functional data observed on the same grid and has been used extensively by our collaborators. We have a long track-record of releasing functional data analysis software and the final form of the function will be part of the next release of *refund*.

References

Besse, P., Ramsay, J.O.: Principal components analysis of sampled functions. Psychometrika **51**, 285–311 (1986)

Besse, P., Cardot, H., Ferraty, F.: Simultaneous nonparametric regressions of unbalanced longitudinal data. Comput. Stat. Data Anal. **24**, 255–270 (1997)

Cai, T., Yuan, M.: Nonparametric Covariance Function Estimation for Functional and Longitudinal Data. Technical report, University of Pennsylvania, Philadelphia, PA (2012)

Cederbaum, J., Pouplier, M., Hoole, P., Greven, S.: Functional linear mixed models for irregularly or sparsely sampled data. Stat. Model. **16**(1), 67–88 (2016)

Chen, H., Wang, Y.: A penalized spline approach to functional mixed effects model analysis. Biometrics **67**(3), 861–870 (2011)

de Boor, C.: A Practical Guide to Splines. Springer, Berlin (1978)

Diggle, P., Heagerty, P., Liang, K.-Y., Zeger, S.: Analysis of Longitudinal Data. Oxford University Press, Oxford (1994)

Durban, M., Harezlak, J., Wand, M.P., Carroll, R.J.: Simple fitting of subject-specific curves for longitudinal data. Stat. Med. **24**(8), 1153–1167 (2005)

Eilers, P., Marx, B.: Flexible smoothing with B-splines and penalties (with discussion). Stat. Sci. **11**, 89–121 (1996)

Eilers, P., Marx, B.: Multivariate calibration with temperature interaction using two-dimensional penalized signal regression. Chemom. Intell. Lab. Syst. **66**, 159–174 (2003)

Fan, J., Gijbels, I.: Local Polynomial Modelling and its Applications. Chapman & Hall, London (1996)

Goldsmith, J., Bobb, J., Crainiceanu, C., Caffo, B., Reich, D.: Penalized functional regression. J. Comput. Graph. Stat. **20**, 830–851 (2010)

Goldsmith, J., Greven, S., Crainiceanu, C.: Corrected confidence bands for functional data using principal components. Biometrics **69**(1), 41–51 (2013)

Huang, L., Scheipl, F., Goldsmith, J., Gellar, J., Harezlak, J., Mclean, M., Swihart, B., Xiao, L., Crainiceanu, C., Reiss, P., Chen, Y., Greven, S., Huo, L., Kundu, M., Wrobel, J.: R package *mgcv*: Methodology for regression with functional data (version 0.1-13). https://cran.r-project.org/web/packages/refund/index.html (2015)

James, G., Hastie, T., Sugar, C.: Principal component models for sparse functional data. Biometrika **87**, 587–602 (2000)

Kaslow, R.A., Ostrow, D.G., Detels, R., Phair, J.P., Polk, B.F., Rinaldo, C.R.: The multicenter aids cohort study: rationale, organization, and selected characteristics of the participants. Am. J. Epidemiol. **126**(2), 310–318 (1987)

Kneip, A.: Nonparametric estimation of common regressors for similar curve data. Ann. Stat. **22**, 1386–1427 (1994)

Peng, J., Paul, D.: A geometric approach to maximum likelihood estimation of functional principal components from sparse longitudinal data. J. Comput. Graph. Stat. **18**, 995–1015 (2009)

Ramsay, J., Dalzell, C.J.: Some tools for functional data analysis (with discussion). J. R. Stat. Soc. B **53**, 539–572 (1991)

Reiss, P.T., Todd Ogden, R.: Smoothing parameter selection for a class of semiparametric linear models. J. R. Stat. Soc. Ser. B (Stat. Methodol.) **71**(2), 505–523 (2009)

Reiss, P., Huang, L., Mennes, M.: Fast function-on-scalar regression with penalized basis expansions. Int. J. Biostat. **6**, 28 (2010)

Rodríguez-Álvarez, M.X., Lee, D.-J., Kneib, T., Durbán, M., Eilers, P.: Fast smoothing parameter separation in multidimensional generalized p-splines: the sap algorithm. Stat. Comput. **25**(5), 941–957 (2015)

Rodríguez-Álvarez, M. X., Durbán, M., Lee, D.-J., Eilers, P.: Fast estimation of multidimensional adaptive P-spline models. http://arxiv.org/pdf/1610.06861.pdf (2016)

Ruppert, D., Wand, M., Carroll, R.: Semiparametric Regression. Cambridge University Press, Cambridge (2003)

Scheipl, F., Staicu, A.-M., Greven, S.: Functional additive mixed models. J. Comput. Graph. Stat. **24**(2), 477–501 (2015)

Seber, G.: A Matrix Handbook for Statisticians. Wiley-Interscience, Hoboken (2007)

Staniswalis, J., Lee, J.: Nonparametric regression analysis of longitudinal data. J. Am. Stat. Assoc. **93**, 1403–1418 (1998)

Wood, S.: Thin plate regression splines. J. R. Stat. Soc. B **65**, 95–114 (2003)

Wood, S.N.: Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. J. R. Stat. Soc. Ser. B (Stat. Methodol.) **73**(1), 3–36 (2011)

Wood, S.: R package *mgcv*: mixed GAM computation vehicle with GCV/AIC/REML, smoothese estimation (version 1.7-24). http://cran.r-project.org/web/packages/mgcv/index.html (2013)

Wood, S.N., Fasiolo, M.: A generalized Fellner–Schall method for smoothing parameter optimization with application to tweedie location, scale and shape models. Biometrics (2017) doi:10.1111/biom.12666

Xiao, L., Li, Y., Ruppert, D.: Fast bivariate P-splines: the sandwich smoother. J. R. Stat. Soc. B **75**, 577–599 (2013)

Xiao, L., Huang, L., Schrack, J., Ferrucci, L., Zipunnikov, V., Crainiceanu, C.: Quantifying the life-time circadian rhythm of physical activity: a covariate-dependent functional approach. Biostatistics **16**, 352–367 (2015)

Xiao, L., Ruppert, D., Zipunnikov, V., Crainiceanu, C.: Fast covariance function estimation for high-dimensional functional data. Stat. Comput. **26**, 409–421 (2016)

Xiao, L., Li, C., Checkley, W., Crainiceanu, C.: R package *face*: fast covariance estimation for sparse functional data (version 0.1-3). https://cran.r-project.org/web/packages/face/index.html (2017)

Xu, G., Huang, J.: Asymptotic optimality and efficient computation of the leave-subject-out cross-validation. Ann. Stat. **40**, 3003–3030 (2012)

Yao, F., Müller, H., Clifford, A., Dueker, S., Follett, J., Lin, Y., Buchholz, B., Vogel, J.: Shrinkage estimation for functional principal component scores with application to the population kinetics of plasma folate. Biometrics **20**, 852–873 (2003)

Yao, F., Müller, H., Wang, J.: Functional data analysis for sparse longitudinal data. J. Am. Stat. Assoc. **100**, 577–590 (2005)