Fast Covariance Estimation for High-dimensional Functional Data

David Ruppert

Apr 7, 2015

Joint work with

- Luo Xiao, postdoc, going to NC State in Fall
  - Primary contributor to this work
- Vadim Zipunnikov, assistant professor
- Ciprian Crainiceanu, professor
  - Is encountering huge data sets.
  - Once gave a presentation "My first 100 Tb of data."
  - Major interest in fast computations with "big data."

All at Johns Hopkins University, Biostatistics.

Outline

- Example: EEG from Sleep Heart Health Study
- Functional Data
- Splines
- Sandwich Smoother
- FACE (FAst Covariance Estimator)
- SVDS (Singular Value Decomposition Smoothed)
- Return to Example
- Simulation Study
- R implementations

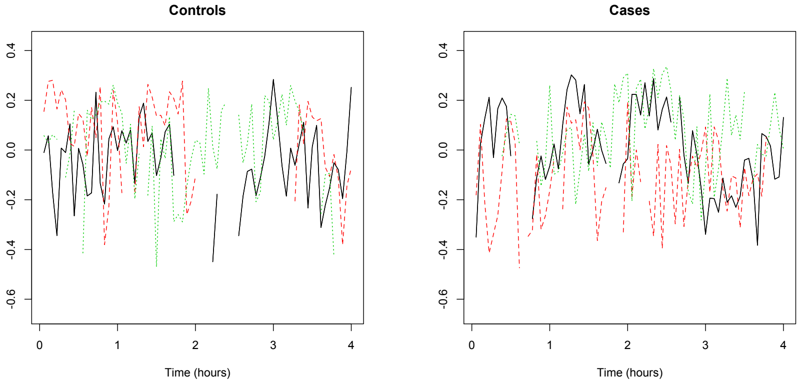
Sleep Heart Health Study (SHHS)

- Large-scale study of sleep and its association with health outcomes.
- Thousands of subjects underwent two in-home polysomnograms at multiple visits.
    - Includes EEG at 125 observations/second.
- δ-power, a summary measure, recorded at 5-second intervals for 4 hours.
    - So we have functional data.
- $12/\text{minute} \times 60 \text{ minutes/hour} \times 4 \text{ hours} = 2880$ measurements/function.

Matched Pairs and Missing Data

- There were 50 matched pairs of controls and sleep apnea cases.
- There are periods of missing data when subjects are awake.

Mean-Centered Curves for Three Matched Pairs



Suppose we observe $Y_i(t)$ on the ith subject, $i = 1, \ldots, n$.

- This function is defined on some interval, say, [0,1].
- The observations are on a fine grid, $t_1, \ldots, t_J$.
  - $t_j = (j-1)/(J-1)$.
- $J = 2880$ in our example.
- The observations are the sum of a signal and noise so we see $Y_i(t) = X_i(t) + \epsilon_i(t)$.
  - X_i is the function of interest.
  - ϵ_i is white noise, that is, uncorrelated, and has variance σ^2.

Mean function: $\mu(t) = E\{X_i(t)\}$

Covariance function of X: $K(s,t) = \operatorname{cov}\{X_i(s), X_i(t)\}$

Covariance function of Y:

$\operatorname{cov}\{Y_i(s), Y_i(t)\} = K(s,t) + \sigma^2 I(s=t)$

- If J is large: can ignore σ^2.

Topic of this talk: Estimation of $K(\cdot,\cdot)$

Let $\widehat{\mu}$ be a smooth estimate of μ.

Let $\widehat{\mathbf{K}}$ be the $J \times J$ sample covariance matrix.

$$\widehat{\mathbf{K}} = n^{-1} \sum_{i=1}^{n} \mathbf{Y}_i \mathbf{Y}_i^{\mathsf{T}}$$

where $\mathbf{Y}_i = \Big(\{ Y_i(t_1) - \widehat{\mu}(t_1) \}, \ldots, \{ Y_i(t_J) - \widehat{\mu}(t_J) \} \Big)^{\mathsf{T}}$.

Problems with High-dimensional Functional Data

- $\widehat{\mathbf{K}}$ is singular if $J > n$.
- For large J, $\widehat{\mathbf{K}}$ may not fit in a computer's memory.
- Smoothing $\widehat{\mathbf{K}}$ to remove noise can be very computationally intensive.

Solution: PCA: Extract relatively few principal components (eigenvectors).

- The PC's may be of interest in themselves.
- They are also used, for example, in functional regression.

Estimating the Eigenvectors

Because of the noise, some type of smoothing is needed in conjunction with PCA. Three possible approaches:

1. smooth the principal components of the sample covariance matrix
2. smooth the sample covariance matrix before performing PCA
3. smooth the Y_i and use the sample covariance of $\widehat{Y}_i$

Our approach connects 2 and 3.

- In fact, they are equivalent under our approach.

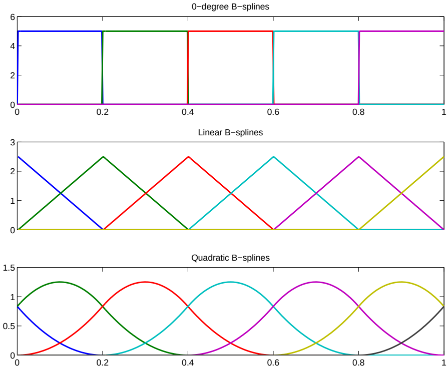
Smoothing the Sample Covariance Matrix

- We can apply a bivariate smoother to $\widehat{\mathbf{K}}$.
- Until recently, no bivariate smoother could handle $J > 500$.
- The recently introduced sandwich smoother can smooth a 500×500 matrix but is not adapted to handle, say, $J = 10,000$.
- This talk introduces FACE = FAst Covariance Estimator.
- FACE is a implementation of the sandwich smoother designed for high-dimensional covariance matrices.

The sandwich smoother and FACE are both spline estimators.

- Splines are piecewise polynomial.
- The polynomial form changes at knots.
- With the degree and knots fixed, splines form a vector space.
- B-splines are a numerically stable basis.

B-splines of Degree 0, 1, and 2



Each B-spline is a different color.

Assume: $y_i = m(x_i) + \epsilon_i$

- Model: $m(x) = \sum_{k=1}^{c} \beta_k B_k(x)$
  - β_k, $k = 1, \ldots, c$, are spline coefficients
  - $(B_1(x), \ldots, B_c(x))$ is a B-spline basis
- Estimate by penalized least-squares.
- The penalty is $\lambda\mathcal{P}$.
- $\mathcal{P}$ is a difference penalty such as
  - $\mathcal{P} = \sum_{k=2}^{c}(\beta_k - \beta_{k-1})^2$ [first differences], or
  - $\mathcal{P} = \sum_{k=3}^{c}\{(\beta_k - \beta_{k-1}) - (\beta_{k-1} - \beta_{k-2})\}^2$ [second differences].
- λ is a smoothing parameter.

The penalized least-squares estimate is
$\widehat{\beta} = (\mathbf{B}^{\mathsf{T}}\mathbf{B} + \lambda\mathbf{D}^{\mathsf{T}}\mathbf{D})^{-1}\mathbf{B}^{\mathsf{T}}\mathbf{y}$.

- $\mathbf{y}$ is the vector of y_i's.
- $\mathbf{B}$ is the "design matrix."
  - The i, jth element of $\mathbf{B}$ is the jth basis function evaluated at x_i.
- $\mathbf{D}$ is a matrix such that $\mathbf{D}\beta$ contains the differences (of the chosen order) of the vector β.
  - So $\mathcal{P} = \boldsymbol{\beta}^{\mathsf{T}}\mathbf{D}^{\mathsf{T}}\mathbf{D}\boldsymbol{\beta}$.
- $\mathbf{S} = \mathbf{B}(\mathbf{B}^{\mathsf{T}}\mathbf{B} + \lambda\mathbf{D}^{\mathsf{T}}\mathbf{D})^{-1}\mathbf{B}^{\mathsf{T}}$ is the smoother (or hat) matrix.
  - The vector of fitted values is $\widehat{\mathbf{y}} = \mathbf{S}\mathbf{y}$.

Bivariate Regression: $y_i = m(s_i, t_i) + \epsilon_i$

Tensor product spline: $m(s, t) = \sum_{k=1}^{c} \sum_{\ell=1}^{c} \beta_{k,l} B_k(s) B_\ell(t)$

- $B_1, \ldots, B_c$ is a univariate basis
- In some applications, one needs different bases in the two variables.
- Using the same basis is appropriate for covariance functions where the two variables are the same.

Estimation is by penalized least squares.

Eilers and Marx's Bivariate P-splines: Penalties

Eilers and Marx's bivariate P-spline uses row penalties and column penalties:

- Arrange the $\beta_{k,\ell}$ in a matrix.
- The row penalty is a univariate difference penalty applied to each row and then summed over rows.
- The column penalty is analogous.

Penalty can be written as

$\beta^{\mathsf{T}}(\lambda_1 \mathbf{I} \otimes \mathbf{D}^{\mathsf{T}}\mathbf{D} + \lambda_2 \mathbf{D}^{\mathsf{T}}\mathbf{D} \otimes \mathbf{I})\beta$ where β is the column vector of the $\beta_{k,\ell}$.

Suppose the y_{ij} are observed on a $J_1 \times J_2$ rectangular grid, e.g., a covariance matrix where $J_1 = J_2 = J$.

- Put the $y_{i,j}$ in a matrix $\mathbf{Y}$.
- The sandwich smoother is $\widehat{\mathbf{Y}} = \mathbf{S}_1 \mathbf{Y} \mathbf{S}_2$. (Xiao et al., 2013, *JRSS-B*).
- Here $\mathbf{S}_1$ and $\mathbf{S}_2$ are univariate smoother matrices.
- Luo discovered the sandwich smoother when studying the asymptotic behavior of the Eilers-Marx bivariate smoother.
- By modifying the Eilers-Marx penalty, the spline estimator was easier to study.
    - The estimator with the modified penalty is equivalent to the sandwich smoother.

Two Representations of the Sandwich Smoother

Sandwich smoother in matrix notation:

$$\widehat{\mathbf{Y}} = \mathbf{S}_1 \mathbf{Y} \mathbf{S}_2.$$

where $\widehat{\mathbf{Y}}$ and $\mathbf{Y}$ are rectangular matrices.

Sandwich smoother in vector notation:

$$\widehat{\mathbf{y}} = (\mathbf{S}_2 \otimes \mathbf{S}_1)\mathbf{y}$$

where $\mathbf{y} = \mathrm{vec}(\mathbf{Y})$ and $\widehat{\mathbf{y}} = \mathrm{vec}(\widehat{\mathbf{Y}})$.

$$
\begin{aligned}
\mathbf{S}_2 \otimes \mathbf{S}_1 &= \left\{\mathbf{B}_2(\mathbf{B}_2^T\mathbf{B}_2 + \lambda_2\mathbf{D}_2^T\mathbf{D}_2)^{-1}\mathbf{B}_2^T\right\} \\
&\otimes \left\{\mathbf{B}_1(\mathbf{B}_1^T\mathbf{B}_1 + \lambda_1\mathbf{D}_1^T\mathbf{D}_1)^{-1}\mathbf{B}_1^T\right\} \\
&= (\mathbf{B}_2 \otimes \mathbf{B}_1)\{\mathbf{B}_2^T\mathbf{B}_2 \otimes \mathbf{B}_1^T\mathbf{B}_1 + \lambda_1\mathbf{B}_2^T\mathbf{B}_2 \otimes \mathbf{D}_1^T\mathbf{D}_1 \\
&+ \lambda_2\mathbf{D}_2^T\mathbf{D}_2 \otimes \mathbf{B}_1^T\mathbf{B}_1 + \lambda_1\lambda_2\mathbf{D}_2^T\mathbf{D}_2 \otimes \mathbf{D}_1^T\mathbf{D}_1\}^{-1}(\mathbf{B}_2 \otimes \mathbf{B}_1)^T.
\end{aligned}
$$

The contributions due to the penalty are in red.

Comparison of the Penalties

For simplicity, assume $\mathbf{D}_1 = \mathbf{D}_2 = \mathbf{D}$.

Eilers-Marx penalty matrix:

$$\lambda_1 \mathbf{I} \otimes \mathbf{D}^\mathsf{T}\mathbf{D} + \lambda_2 \mathbf{D}^\mathsf{T}\mathbf{D} \otimes \mathbf{I}$$

Sandwich smoother penalty matrix:

$$\mathbf{P} = \lambda_1 \mathbf{B}_2^T \mathbf{B}_2 \otimes \mathbf{D}^T\mathbf{D} + \lambda_2 \mathbf{D}^T\mathbf{D} \otimes \mathbf{B}_1^T \mathbf{B}_1 + \lambda_1\lambda_2 \mathbf{D}^T\mathbf{D} \otimes \mathbf{D}^T\mathbf{D}$$

Sandwich Smoother for Covariance Matrices

Let $\widehat{\mathbf{K}}$ be the sample covariance matrix. Recall that

$$\widehat{\mathbf{K}} = \sum_{i=1}^{n} \mathbf{Y}_i \mathbf{Y}_i^{\mathsf{T}} \tag{1}$$

where $\mathbf{Y}_i = \{y_i(t_1) - \widehat{\mu}(t_1), \ldots, y_i(t_J) - \widehat{\mu}(t_J)\}^{\mathsf{T}}$.

Applying the sandwich smoother to $\widehat{\mathbf{K}}$, we obtain

$$\widetilde{\mathbf{K}} = \mathbf{S}\widehat{\mathbf{K}}\mathbf{S}. \tag{2}$$

Substituting (1) into (2), we obtain

$$\widetilde{\mathbf{K}} = \sum_{i=1}^{n} (\mathbf{S}\mathbf{Y}_i)(\mathbf{S}\mathbf{Y}_i)^{\mathsf{T}}.$$

Recall the three possible approaches to smooth PCA:

1. smooth the principal components of the sample covariance matrix
2. smooth the sample covariance matrix before performing PCA
3. smooth the Y_i and use the sample covariance of $\widehat{Y}_i$

From previous frame:

$$\widetilde{\mathbf{K}} = \sum_{i=1}^{n} (\mathbf{S}\mathbf{Y}_i)(\mathbf{S}\mathbf{Y}_i)^{\mathsf{T}}.$$

With the sandwich smoother approaches, 2 and 3 are equivalent.

The Sandwich Smoother is Fast

- The sandwich smoother was discovered while studying the asymptotic distribution of bivariate spline estimators.
- It was soon realized that the sandwich estinator could speed computations considerably.
- For fixed smoothing parameters, a bivariate spline can be computed as a generalized linear array model (GLAM) (Currie, Durban, and Eilers, 2006, JRSS-B)
- The bottleneck was in computing the effective degrees of freedom (DF) needed for GCV (Generalized Cross Validtion) to select the smoothing parameters.

The fitted values are computed as

$$\widehat{\mathbf{y}} = \mathbf{S}\mathbf{y}.$$

Here $\mathbf{y}$ and $\widehat{\mathbf{y}}$ are vectors and $\mathbf{S}$ is the smoother matrix.

The degrees of freedom of the smoother is defined as

$$\mathrm{DF} = \mathrm{tr}(\mathbf{S}).$$

For OLS: $\mathrm{tr}(\mathbf{S})$ equals the dimension of the model when there is no penalty.

PenLS: $\mathrm{tr}(\mathbf{S}) <$ dimension of model.

- Because of the penalty, there are effectively less parameters.

DF for the Sandwich Smoother

Recall: Sandwich smoother in vector notation:

$$\widehat{\mathbf{y}} = (\mathbf{S}_2 \otimes \mathbf{S}_1)\mathbf{y},$$

where $\mathbf{y} = \mathrm{vec}(\mathbf{Y})$ and $\widehat{\mathbf{y}} = \mathrm{vec}(\widehat{\mathbf{Y}})$.

From the vector notation, we see that DF $= \mathrm{tr}(\mathbf{S}_2 \otimes \mathbf{S}_1) = \mathrm{tr}(\mathbf{S}_2)\mathrm{tr}(\mathbf{S}_1)$: fast to compute

The smoother matrix of a univariate spline can be diagonalized

$$\mathbf{S} = \mathbf{O}\,\mathsf{diag}\{(1 + \lambda\kappa_k)^{-1}\}\mathbf{O}^{\mathsf{T}}$$

where $\mathbf{O}$ is $n \times c$, has orthogonal columns, and does not depend on λ, so

$$\mathsf{tr}(\mathbf{S}) = \mathsf{tr}\left\{\mathsf{diag}\{(1 + \lambda\kappa_k)^{-1}\}\mathbf{O}\mathbf{O}^{\mathsf{T}}\right\} = \sum_{k=1}^{c}(1 + \lambda\kappa_k)^{-1}.$$

The diagonalization is performed once and then this sum computes $\mathsf{tr}(\mathbf{S})$ for all values of (λ_1, λ_2).

The sandwich estimator applies this method to $\mathbf{S}_1$ and $\mathbf{S}_2$.

J^2	$c_1 c_2$	Sandwich smoother	E-M/GLAM	TPRS
20^2	10^2	0.06(0.24)	4.09(19.74)	0.53
40^2	20^2	0.08(0.30)	94.76(344.13)	19.50
80^2	35^2	0.13(0.45)	1379.21(5487.33)	1032.07
300^2	42^2	0.18(0.58)	3798.23(15192.92)	–
500^2	57^2	0.32(0.89)	21023.44(84093.76)	–

For the sandwich smoother and E-M/GLAM, the times are for a 20×20 (40×40) grid of λ. The covariance matrix is $J \times J$. c_i is the dimension of the ith basis.

TPRS = thin-plate regression spline using `bam()` (faster version of `gam()`) in the `mgcv` package.

Recall the sandwich smoother of the sample covariance matrix:

$$\widetilde{\mathbf{K}} = \mathbf{S}\widehat{\mathbf{K}}\mathbf{S}.$$

All four matrices are $J \times J$.

The rank of $\widetilde{\mathbf{K}}$ is at most $\min(J, c)$ where

- c is the dimension of the spline basis so $c = \text{rank}(\mathbf{S})$.

We are interested in the case where $c \ll J$.

- Then most of the eigenvalues of $\widetilde{\mathbf{K}}$ are 0.
- We want an efficient method to find the non-zero eigenvalues and their eigenvectors.

Start with an eigen-decomposition:
$(\mathbf{B}^\mathsf{T}\mathbf{B})^{-1/2}\mathbf{P}(\mathbf{B}^\mathsf{T}\mathbf{B})^{-1/2} = \mathbf{U}\text{diag}(\mathbf{s})\mathbf{U}^\mathsf{T}$. ($c \times c$ matrices.)

Then define $\mathbf{A}_S = \mathbf{B}(\mathbf{B}^\mathsf{T}\mathbf{B})^{-1/2}\mathbf{U}$. (Has orthogonal columns and does not depend on λ.)

Then the smoother matrix can be represented as
$\mathbf{S} = \mathbf{A}_S\mathbf{\Sigma}_S\mathbf{A}_S^\mathsf{T}$ where $\mathbf{\Sigma}_S = \{\mathbf{I}_c + \lambda\text{diag}(\mathbf{s})\}^{-1}$ ($c \times c$).

$\widetilde{\mathbf{K}} = \mathbf{S}\widehat{\mathbf{K}}\mathbf{S} = \mathbf{A}_S\left(n^{-1}\mathbf{\Sigma}_S\widetilde{\mathbf{Y}}\widetilde{\mathbf{Y}}^\mathsf{T}\mathbf{\Sigma}_S\right)\mathbf{A}_S^\mathsf{T}$ where $\widetilde{\mathbf{Y}} = \mathbf{A}_S^\mathsf{T}\mathbf{Y}$.

$\left(n^{-1}\mathbf{\Sigma}_S\widetilde{\mathbf{Y}}\widetilde{\mathbf{Y}}^\mathsf{T}\mathbf{\Sigma}_S\right) = \mathbf{A}\mathbf{\Sigma}\mathbf{A}^\mathsf{T}$. (Another $c \times c$ eigendecomposition)

We arrive at the eigendecomposition of $\widetilde{\mathbf{K}}$:
$\widetilde{\mathbf{K}} = (\mathbf{A}_S\mathbf{A})\mathbf{\Sigma}(\mathbf{A}_S\mathbf{A})^\mathsf{T}$.

In the last frame we found the eigendecomposition of $\widetilde{\mathbf{K}}$:

$\widetilde{\mathbf{K}} = (\mathbf{A}_S\mathbf{A})\boldsymbol{\Sigma}(\mathbf{A}_S\mathbf{A})^\mathsf{T}$.

- $\boldsymbol{\Sigma}$ is $c \times c$ and diagonal.
    - The diagonal elements are the nontrivial eigenvalues of $\widetilde{\mathbf{K}}$.
- $(\mathbf{A}_S\mathbf{A})$ is $J \times c$ with orthogonal columns.
    - These are the associated eigenvectors of $\widetilde{\mathbf{K}}$.

Selecting the Smoothing Parameters

Pooled generalized cross validation: Select λ by minimizing

$$\mathrm{PGCV}(\lambda) = \frac{\sum_{j=1}^{J} \|\mathbf{Y}_i - \mathbf{SY}_i\|^2}{\{1 - \mathsf{tr}(\mathbf{S})/J\}^2}$$

We have developed a fast method to compute PGCV(λ).

Let $\psi_1(t), \ldots, \psi_N(t)$ be the eigenvectors extracted by PCA. Then the truncated Karhunen-Loève decompositon is

$$X_i = \sum_{k=1}^{N} \xi_{i,k} \psi_k(t)$$

where the score $\xi_{i,k}$ is given by

$$\xi_{i,k} = \int X_i(t) \psi_i(t) \, dt.$$

- We have develop fast methods to estimate the scores by numerical integration or BLUPs.

When data are missing, we alternate between

- Smoothing with FACE
- Prediction of missing values (imputation)

SVDS = Singular Value Decomposition with Smoothing.

Let $\mathbf{Y}$ be the $J \times n$ data matrix.

$$\underbrace{\mathbf{Y}}_{J\times n} = \underbrace{\mathbf{U}}_{J\times n}\underbrace{\mathbf{D}}_{n\times n}\underbrace{\mathbf{V}}_{n\times n}{}^{\mathsf{T}}.$$

- The columns of $\mathbf{Y}$ are the mean-centered functions.
- $\mathbf{U}$ has orthogonal columns.
- $\mathbf{D}$ is diagonal.
- $\mathbf{V}$ is orthogonal.

From previous slide:

$$\underbrace{\mathbf{Y}}_{J\times n} = \underbrace{\mathbf{U}}_{J\times n}\underbrace{\mathbf{D}}_{n\times n}\underbrace{\mathbf{V}}_{n\times n}{}^{\mathsf{T}}.$$

- The columns of $\mathbf{U}$ contain the eigenvectors of $\widehat{K}$.
  - These are smoothed by penalized splines.
- The diagonal elements of $\mathbf{D}$ contain the square-roots of the non-zero eigenvalues.
  - These are squared.

We return to the SHHS example.

The model for the two curves for the ith matched pair is:

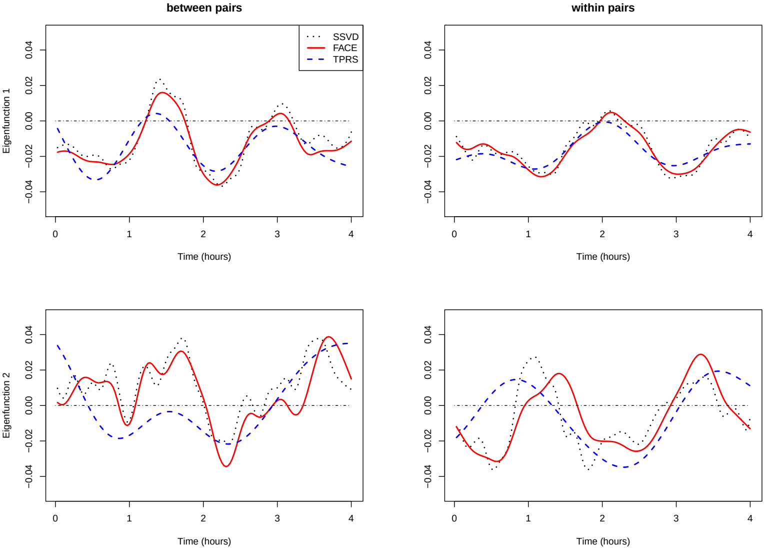
$$Y_{iA}(t) = \mu_A(t) + X_i(t) + U_{iA}(t) + \epsilon_{iA}(t)$$
$$Y_{iC}(t) = \mu_C(t) + X_i(t) + U_{iC}(t) + \epsilon_{iC}(t)$$

- "A" $=$ apnea and "C" $=$ control.
- μ_A and μ_B are the mean curves.
- $X_i(t)$ captures the between-subjects correlation and has correlation function $K_X(\cdot,\cdot)$.
- U_{iA} and U_{iC} are independent and have covariance $K_U(\cdot,\cdot)$.

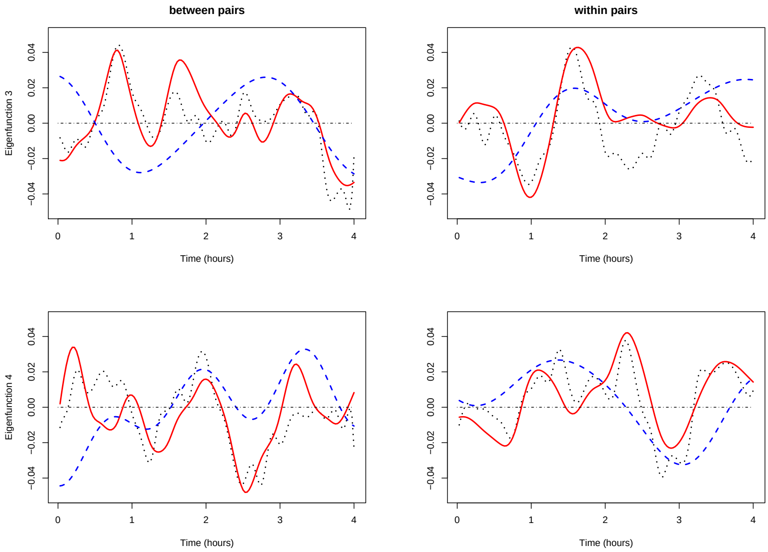
We applied three estimators to the SHHS data:

- Thin-plate regression spline (TPRS) using `bam()` in the `mgcv` package.
  - Only 35 knots.
  - Running time was 3 hours.
- FACE
  - 100 knots
  - less than 10 seconds
- SVDS

Estimates of Eigenvectors 1 and 2



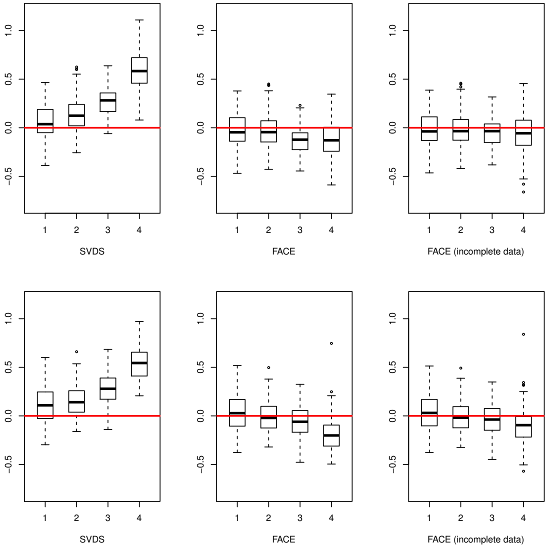
Estimates of Eigenvectors 3 and 4



Simulations Setup

- $J = 3,000$
- $n = 50$
- eigenfunctions were:
  $\left\{ \sqrt{2}\sin(2\pi t), \sqrt{2}\cos(2\pi t), \sqrt{2}\sin(4\pi t), \sqrt{2}\cos(4\pi t) \right\}$
- missing data: Sections of $0.065J$ observations were missing at random.

Simulations: Boxplots of Estimated Eigenvalues



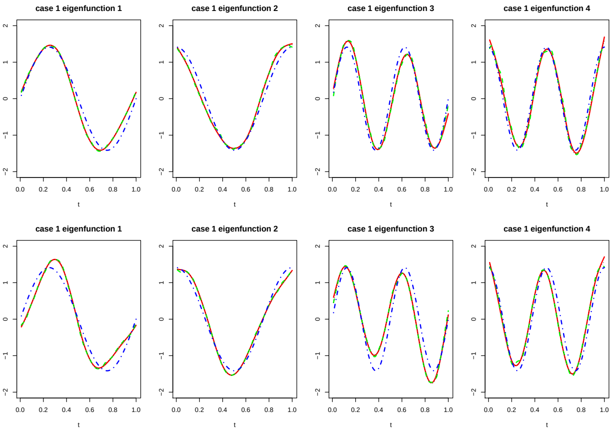
Estimating Eigenvalues: 100 $\times$ Average Squared Errors

Eigenvalue	SVDS	FACE	FACE (incomplete data)
1	4.00	3.39	7.34
2	1.27	0.82	1.61
3	0.62	0.22	0.41
4	0.62	0.07	0.08

Computation Time in Seconds

J	I	FACE 100 knots	FACE 500 knots	SVDS 100 knots	SVDS 500 knots	Sandwich 100 knots	Sandwich 500 knots
$3,000$	50	0.27	8.61	1.40	36.38	86.89	124.78
	500	0.76	13.61	6.28	42.07	93.94	131.82
$5,000$	50	0.48	13.47	2.27	62.99	433.33	467.83
	500	1.37	18.01	8.11	73.16	509.67	570.79
$10,000$	50	0.95	23.88	4.52	114.57	-	-
	500	3.07	35.98	20.91	133.57	-	-

Simulations: Estimated Eigenfunctions



True = Blue; FACE = red; FACE with incomplete data = green; SVDS = Black

In the refund (Regression with Functional Data) package:

- fbps(): sandwich smoother
- fpca.face(): functional PCA by FACE
- pfr(. . ., smooth.option = "fpca.face"): penalized functional regression with the functional covariance represented by the PC basis

Thanks for coming!