Journal of Statistical Software

May 2019, Volume 89, Issue 2. doi: 10.18637/jss.v089.i02

BFDA: A MATLAB Toolbox for Bayesian Functional Data Analysis

Jingjing Yang
Emory University

Peng Ren
Bank of America Corporation

Abstract

We provide a MATLAB toolbox, **BFDA**, that implements a Bayesian hierarchical model to smooth multiple functional data samples with the assumptions of the same underlying Gaussian process distribution, a Gaussian process prior for the mean function, and an Inverse-Wishart process prior for the covariance function. This model-based approach can borrow strength from all functional data samples to increase the smoothing accuracy, as well as simultaneously estimate the mean-covariance functions. An option of approximating the Bayesian inference process using cubic B-spline basis functions is integrated in **BFDA**, which allows for efficiently dealing with high-dimensional functional data. Examples of using **BFDA** in various scenarios and conducting follow-up functional regression are provided. The advantages of **BFDA** include: (1) simultaneously smooths multiple functional data samples and estimates the mean-covariance functions in a nonparametric way; (2) flexibly deals with sparse and high-dimensional functional data with stationary and nonstationary covariance functions, and without the requirement of common observation grids; (3) provides accurately smoothed functional data for follow-up analysis.

Keywords: functional data analysis, Bayesian hierarchical model, Gaussian process, cubic B-spline basis functions, MATLAB.

1. Introduction

Since Ramsay and Dalzell (1991) first coined the term "functional data analysis" (FDA) for analyzing data that are realizations of a continuous function, many statistical methods and tools have been proposed for FDA. For examples, Graves *et al.* (2010) provided both, the R (R Core Team 2019) package **fda** (Ramsay *et al.* 2018) and the MATLAB (The MathWorks Inc. 2017) package **fdaM** (Ramsay 2014) for typical functional data analysis (Ramsay and Silverman 2002, 2005); Febrero-Bande and de la Fuente (2012) provided the R package **fda.usc** for implementing nonparametric functional data analysis methods (Vieu and Ferraty 2006)

with **fda** (Ramsay *et al.* 2018); Yao *et al.* (2005a,b) developed the key technique of functional principal component analysis (FPCA) for analyzing sparse functional data, accompanied by the MATLAB package **PACE** (Yao *et al.* 2015); Crainiceanu and Goldsmith (2010) proposed insights about implementing the standard Bayesian FDA using **WinBUGS** (Sturtz *et al.* 2005); and Shi and Cheng (2014) derived the R package **GPFDA** for applying the Bayesian nonparametric Gaussian process (GP) regression models (Shi and Choi 2011). However, the smoothing step that constructs functions from noisy discrete data has been neglected by most of the existing FDA methods and tools, where functional data representations are integrated in the analyzing models. On the other hand, most of the existing smoothing methods process functional samples individually (e.g., cubic smoothing spline, CSS, and kernel smoothing; Green and Silverman 1993; Ramsay and Silverman 2005), which is likely to induce bias when functional samples are of the same distribution.

Here, we provide a MATLAB toolbox **BFDA** for simultaneously smoothing multiple functional data samples from the same distribution and estimating the underlying mean-covariance functions, using a nonparametric Bayesian hierarchical model (BHM) with Gaussian-Wishart processes (Yang *et al.* 2016). This model-based approach borrows strength through modeling the shared mean-covariance functions, thus producing more accurate smoothing results than the individually smoothing methods. Moreover, **BFDA** is flexible for analyzing sparse and dense functional data without the requirement of common observation grids, suitable for analyzing functional data with both stationary and nonstationary covariance functions, and efficient for high-dimensional functional data using an efficient Bayesian approximating reference algorithm with Basis functions (BABF; Yang *et al.* 2017). In addition, **BFDA** provides options for implementing the standard Bayesian GP regression method, conducting Bayesian principal component analysis, applying cubic smoothing splines per functional sample, and using the **fdaM** package (Ramsay 2014) for follow-up analysis.

In the following context, we first review the BHM and BABF methods in Section 2, and then provide examples using **BFDA** with simulated data in Section 3. In Section 4, we compare the functional linear regression results by **fdaM** (Ramsay 2014) using the smoothed data by **BFDA** and CSS. Last, we conclude with a discussion in Section 5. Details of input options and outputs are provided in Appendix A. Example MATLAB scripts for generating the simulation results in this paper are contained in the supplementary material.

2. Methods overview

2.1. Model-based Bayesian smoothing method: BHM

Consider a total of n functional data samples $\{X_i(t); t \in \mathcal{T}, i = 1, 2, \ldots, n\}$ that are generated from the same stochastic process with independent measurement errors. In order to simultaneously smooth all noisy functional samples and estimate mean-covariance functions, Yang *et al.* (2016) proposed the following Bayesian hierarchical model (BHM) with Gaussian-Wishart processes:

$$X_i(t) = Z_i(t) + \epsilon_i(t);\ Z_i(\cdot) \sim GP(\mu_Z(\cdot), \Sigma_Z(\cdot,\cdot)),\ \epsilon_i(\cdot) \sim N(0, \sigma_\epsilon^2); \tag{1}$$

$$\mu_Z(\cdot)|\Sigma_Z(\cdot,\cdot) \sim GP\left(\mu_0(\cdot), \frac{1}{c}\Sigma_Z(\cdot,\cdot)\right),\ \Sigma_Z(\cdot,\cdot) \sim IWP(\delta, \sigma_s^2 A(\cdot,\cdot)),\ \sigma_\epsilon^2 \sim IG(a_\epsilon, b_\epsilon);$$

$$\sigma_s^2 \sim IG(a_s, b_s);$$

where $\{Z_i(t); i = 1, \ldots, n\}$ denotes the underlying true functional data following the same GP distribution with mean function $\mu_Z(\cdot)$ and covariance function $\Sigma_Z(\cdot, \cdot)$, *IWP* denotes the Inverse-Wishart process (IWP) prior (Dawid 1981) for the covariance function, *IG* denotes the Inverse-Gamma prior, and $(\mu_0(\cdot), c, \delta, A(\cdot, \cdot), a_\epsilon, b_\epsilon, a_s, b_s)$ are hyper-parameters to be determined. The IWP prior on $\Sigma_Z(\cdot, \cdot)$ models the covariance function nonparametrically and therefore makes the BHM method suitable for analyzing functional data with unknown stationary and nonstationary covariance structures. The hyper-parameter σ_s^2 provides the flexibility of estimating the scale of the covariance structure in the IWP prior from the data.

For the hyper-parameter setup, we take $\mu_0(\cdot)$ as the smoothed empirical mean estimate, c as 1 for the same prior data variation on the mean function as on the functional data, δ as 5 for a non-informative prior on the variance function, and determine $(a_\epsilon, b_\epsilon, a_s, b_s)$ by matching the hyper-prior moments with the empirical estimates. In addition, $A(\cdot, \cdot)$ can be taken as the Matérn correlation kernel for analyzing functional data with stationary covariance (default in **BFDA**),

$$\text{Matern}_{\text{cor}}(d; \rho, \nu) = \frac{1}{\Gamma(\nu) 2^{\nu-1}} \left(\sqrt{2\nu}\frac{d}{\rho}\right)^{\nu} K_\nu \left(\sqrt{2\nu}\frac{d}{\rho}\right), \quad d \geq 0,\ \rho > 0,\ \nu > 0,$$

where d denotes the distance between two grid points, ρ is the scale parameter, ν is the order of smoothness, and $K_\nu(\cdot)$ is the modified Bessel function of the second kind. Particularly, $A(\cdot, \cdot)$ can also be taken as a smoothed empirical covariance estimate for analyzing functional data with nonstationary covariance.

Although the BHM is constructed with infinite-dimensional Gaussian-Wishart processes, practical posterior inference will be conducted in a finite manner, e.g., on the observation grids $\{\boldsymbol{t}_i\}$, the pooled grid $\boldsymbol{t} = \cup_{i=1}^n \boldsymbol{t}_i$, or other user-specified evaluation grids. For accommodating uncommon observation grids, especially sparsely observed data, BHM evaluates data functions and mean-covariance functions on the pooled grid, while estimating the unobserved functional data by conditioning on the observations (similar approach as used by **PACE**).

For simplified notation, we denote $X_i(\boldsymbol{t}_i)$ by $\boldsymbol{X}_{\boldsymbol{t}_i}$, $Z_i(\boldsymbol{t}_i)$ by $\boldsymbol{Z}_{\boldsymbol{t}_i}$, $\mu_0(\boldsymbol{t})$ by $\boldsymbol{\mu}_0$, $\mu_Z(\boldsymbol{t})$ by $\boldsymbol{\mu}_Z$, $\Sigma_Z(\boldsymbol{t}, \boldsymbol{t})$ by $\boldsymbol{\Sigma}_Z$, and $A(\boldsymbol{t}, \boldsymbol{t})$ by $\boldsymbol{A}$. BHM conducts Bayesian inference for $(\{\boldsymbol{Z}_{\boldsymbol{t}_i}\}, \boldsymbol{\mu}_Z, \boldsymbol{\Sigma}_Z, \sigma_\epsilon^2, \sigma_s^2)$ by a Monte Carlo Markov chain (MCMC) algorithm (essentially a Gibbs sampler; Geman and Geman 1984) as follows (refer to Yang *et al.* 2016 for more details):

- Step 0: Set initial values. Set hyper-parameters $(c, \boldsymbol{\mu_0}, \nu, \rho, a_\epsilon, b_\epsilon, a_s, b_s)$. Take $(\boldsymbol{\mu}, \sigma_\epsilon^2)$ as respective empirical estimates, $\{\boldsymbol{Z}_{\boldsymbol{t}_i}\}$ as the raw data $\{\boldsymbol{X}_{\boldsymbol{t}_i}\}$, and $\boldsymbol{\Sigma}_Z$ as an identity matrix.

- Step 1: Conditioning on $\{\boldsymbol{X}_{\boldsymbol{t}_i}\}$ and $(\boldsymbol{\mu}_Z, \boldsymbol{\Sigma}_Z)$, update $\{\boldsymbol{Z}_{\boldsymbol{t}_i}\}$ from the corresponding conditional multivariate normal (MVN) distributions.

- Step 2: Conditioning on $\{\boldsymbol{X}_{\boldsymbol{t}_i}\}$ and $\{\boldsymbol{Z}_{\boldsymbol{t}_i}\}$, update σ_ϵ^2 from the conditional Inverse-Gamma (IG) distribution.

- Step 3: Conditional on $\{\boldsymbol{Z}_{\boldsymbol{t}_i}\}$ and $\boldsymbol{\Sigma}_Z$, update $\boldsymbol{\mu}_Z$ from the conditional MVN distribution.

- Step 4: Conditioning on $\{\boldsymbol{Z}_{\boldsymbol{t}_i}\}$ and $\boldsymbol{\mu}_Z$, update $\boldsymbol{\Sigma}_Z$ from the conditional Inverse-Wishart (IW) distribution.

- Step 5: Conditioning on $\boldsymbol{\Sigma}_Z$, update σ_s^2 from the conditional Gamma distribution.

Specifically, the averages of posterior samples of $\{\{\boldsymbol{Z}_{\boldsymbol{t}_i}\}, \boldsymbol{\mu}_Z, \boldsymbol{\Sigma}_Z\}$ are taken as estimates for functional signals and mean-covariance functions.

In addition, **BFDA** uses the existing MATLAB package **mcmcdiag** (Särkkä and Aki 2014) to diagnose the MCMC convergence by the potential scale reduction factor (PSRF; Gelman and Rubin 1992), and implements the method proposed by Yuan and Johnson (2012) with the pivotal discrepancy measures (PDM) of standardized residuals for the goodness-of-fit diagnosis of the assumed model.

2.2. Alternative efficient Bayesian inference algorithm: BABF

Because BHM (Yang *et al.* 2016) has computational complexity $O(np^3m)$ with n samples, p pooled-grid points, and m MCMC iterations, the method encounters a computational bottleneck for analyzing functional data with large pooled-grid dimension p. To resolve this computational bottleneck, **BFDA** enables the option of using the alternative efficient Bayesian inference algorithm BABF that was proposed by Yang *et al.* (2017). Essentially, BABF conducts approximating posterior inference of BHM with basis functions, which greatly improves computation efficiency. Here, we briefly outline the BABF algorithm.

To implement the BABF algorithm, one need to first select a working grid based on data density, $\boldsymbol{\tau} = (\tau_1, \tau_2, \ldots, \tau_L)^\top \subset \mathcal{T}$, where the dimension of the working grid is generally much smaller than the pooled observation dimension ($L << p$). Then $\{Z_i(\boldsymbol{\tau})\}$ can be approximated by using a system of basis functions (e.g., cubic B-splines as implemented by **BFDA**). Let $\boldsymbol{B}(\cdot) = [b_1(\cdot), b_2(\cdot), \ldots, b_K(\cdot)]$ denote K selected basis functions with coefficients $\boldsymbol{\zeta}_i = (\zeta_{i1}, \zeta_{i2}, \ldots, \zeta_{iK})^\top$, then $Z_i(\boldsymbol{\tau}) = \sum_{k=1}^K \zeta_{ik} b_k(\boldsymbol{\tau}) = \boldsymbol{B}(\boldsymbol{\tau})\boldsymbol{\zeta}_i$. Generally, K can be selected as equivalent to L, and then the basis function coefficients can be represented by $\boldsymbol{\zeta}_i = \boldsymbol{B}(\boldsymbol{\tau})^{-1} Z_i(\boldsymbol{\tau})$, a linear transformation of $Z_i(\boldsymbol{\tau})$. Note that even if $\boldsymbol{B}(\boldsymbol{\tau})$ is singular or non-square, $\boldsymbol{\zeta_i}$ can still be written as a linear transformation of $Z_i(\boldsymbol{\tau})$ by using the generalized inverse (James 1978) of $\boldsymbol{B}(\boldsymbol{\tau})$.

Because $\boldsymbol{\zeta}_i$ is a linear transformation of $Z_i(\boldsymbol{\tau})$ that follows a MVN distribution under the assumptions in Equation 1, the induced Bayesian hierarchical model for $\{\boldsymbol{\zeta}_i\}$ is derived as

$$\boldsymbol{\zeta}_i \sim MVN(\boldsymbol{\mu}_{\boldsymbol{\zeta}},\ \boldsymbol{\Sigma}_{\boldsymbol{\zeta}});\ \boldsymbol{\mu}_{\boldsymbol{\zeta}} = \boldsymbol{B}(\boldsymbol{\tau})^{-1}\mu_Z(\boldsymbol{\tau});\ \boldsymbol{\Sigma}_{\boldsymbol{\zeta}} = \boldsymbol{B}(\boldsymbol{\tau})^{-1}\Sigma_Z(\boldsymbol{\tau},\boldsymbol{\tau})\boldsymbol{B}(\boldsymbol{\tau})^{-\top}. \tag{2}$$

Further, from the assumed priors of $(\mu_Z(\cdot), \Sigma_Z(\cdot,\cdot))$ in Equation 1, with $\Psi(\boldsymbol{\tau},\boldsymbol{\tau}) = \sigma_s^2 A(\boldsymbol{\tau},\boldsymbol{\tau})$, the following priors of $(\boldsymbol{\mu}_{\boldsymbol{\zeta}}, \boldsymbol{\Sigma}_{\boldsymbol{\zeta}})$ are also induced:

$$\begin{aligned} \boldsymbol{\mu}_{\boldsymbol{\zeta}}|\boldsymbol{\Sigma}_{\boldsymbol{\zeta}} &\sim MVN\left(\boldsymbol{B}(\boldsymbol{\tau})^{-1}\mu_0(\boldsymbol{\tau}),\ c\boldsymbol{\Sigma}_{\boldsymbol{\zeta}}\right); \\ \boldsymbol{\Sigma}_{\boldsymbol{\zeta}} &\sim IW(\delta,\ \boldsymbol{B}(\boldsymbol{\tau})^{-1}\Psi(\boldsymbol{\tau},\boldsymbol{\tau})\boldsymbol{B}(\boldsymbol{\tau})^{-\top}). \end{aligned} \tag{3}$$

Then, the BABF inference algorithm by MCMC has computation complexity $O(nK^3m)$, comparing to $O(np^3m)$ by BHM where $p >> K$. The MCMC steps for BABF are presented as follows (refer to Yang *et al.* (2017) for more details):

- Step 0: Set initial values similarly as in BHM. Set hyper-parameters $(c, \boldsymbol{\mu_0}, \nu, \rho, a_\epsilon, b_\epsilon, a_s, b_s)$. Take $(\mu_Z(\boldsymbol{\tau}), \Sigma_Z(\boldsymbol{\tau}, \boldsymbol{\tau}), \sigma_\epsilon^2)$ as empirical estimates, inducing the initial values for $(\boldsymbol{\mu_\zeta}, \boldsymbol{\Sigma_\zeta})$ by Equation 2.
- Step 1: Conditioning on $\{\boldsymbol{X_{t_i}}\}$ and $(\boldsymbol{\mu_\zeta}, \boldsymbol{\Sigma_\zeta}, \sigma_\epsilon^2)$, update $\{\boldsymbol{\zeta}_i\}$ from the conditional MVN distribution.
- Step 2: Conditioning on $\{\boldsymbol{\zeta}_i\}$, update $\boldsymbol{\mu_\zeta}$ and $\boldsymbol{\Sigma_\zeta}$ respectively from the conditional MVN and IW distributions.
- Step 3: Conditioning on $(\{\boldsymbol{\zeta}_i\}, \boldsymbol{\mu_\zeta}, \boldsymbol{\Sigma_\zeta})$, approximate $\{\boldsymbol{Z_{t_i}}, \mu_Z(\boldsymbol{t}_i), \Sigma_Z(\boldsymbol{t}_i, \boldsymbol{t}_i), \Sigma_Z(\boldsymbol{\tau}, \boldsymbol{t}_i), \Sigma_Z(\boldsymbol{t}_i, \boldsymbol{\tau}), \Sigma_Z(\boldsymbol{\tau}, \boldsymbol{\tau})\}$ by

$$\boldsymbol{Z_{t_i}} = \boldsymbol{B}(\boldsymbol{t}_i)\boldsymbol{\zeta}_i,\ \mu_Z(\boldsymbol{t}_i) = \boldsymbol{B}(\boldsymbol{t}_i)\boldsymbol{\mu_\zeta},\ \Sigma_Z(\boldsymbol{t}_i, \boldsymbol{t}_i) = \boldsymbol{B}(\boldsymbol{t}_i)\boldsymbol{\Sigma_\zeta}\boldsymbol{B}(\boldsymbol{t}_i)^\top,$$

$$\Sigma_Z(\boldsymbol{\tau}, \boldsymbol{t}_i)^\top = \Sigma_Z(\boldsymbol{t}_i, \boldsymbol{\tau}) = \boldsymbol{B}(\boldsymbol{t}_i)\boldsymbol{\Sigma_\zeta}\boldsymbol{B}(\boldsymbol{\tau})^\top,\ \Sigma_Z(\boldsymbol{\tau}, \boldsymbol{\tau}) = \boldsymbol{B}(\boldsymbol{\tau})\boldsymbol{\Sigma_\zeta}\boldsymbol{B}(\boldsymbol{\tau})^\top.$$

- Step 4: Conditioning on $\{\boldsymbol{Z_{t_i}}\}$ and $\{\boldsymbol{X_{t_i}}\}$, update σ_ϵ^2 by the conditional IG distribution.
- Step 5: Conditioning on $\Sigma_Z(\boldsymbol{\tau}, \boldsymbol{\tau})$, update σ_s^2 by the conditional Gamma distribution.

As a result, the posterior estimates of $(\{\boldsymbol{\zeta}_i\}, \boldsymbol{\mu_\zeta}, \boldsymbol{\Sigma_\zeta})$ are given by the averages of the MCMC samples, which are then used to estimate $\{\boldsymbol{Z_{t_i}}, \mu_Z(\boldsymbol{t}_i), \Sigma_Z(\boldsymbol{t}_i, \boldsymbol{t}_i)\}$ by the approximation formulas in Step 3.

BFDA uses the existing MATLAB package **bspline** (Hunyadi 2010) to construct B-spline basis functions that are widely used by GP regression methods (Rasmussen and Williams 2006; Shi *et al.* 2007); and generates the optimal knot sequence for interpolation at the working grid τ with MATLAB function `optknt()` (Gaffney and Powell 1976; Micchelli *et al.* 1976; De Boor 1977). Yang *et al.* (2017) instructed selecting $\boldsymbol{\tau}$ to represent data densities (L may be selected by grid search with test data), such as taking the $\left(\frac{100}{L+1}, \ldots, \frac{L\times 100}{L+1}\right)$ percentiles of the pooled grid, or the equally-spaced grid for evenly distributed data.

3. Examples with simulated data

In this section, we provide examples of using **BFDA** to analyze simulated functional data with stationary and nonstationary covariance functions, common and uncommon (sparse) observation grids, as well as random observation grids. The simulation data used for the example results were generated with `n = 30` functional samples, pooled-grid dimension `p = 40`, functional data observation domain (`au = 0`, `bu` $= \pi/2$), functional data standard variation `s` $= \sqrt{5}$, signal-to-noise ratio `r = 2` (i.e., the ratio between the signal and noise standard deviations), order of smoothness `nu = 3.5` and scale parameter `rho = 0.5` in the Matérn function, `dense = 0.6` proportion of observation points on the pooled grid `pgrid` (equally spaced grid over $(0, \pi/2)$ with length 40).

3.1. Simulate functional data from a Gaussian process

BFDA provides the convenience of generating simulated functional data from a shared GP with mean function $\mu(t) = 3\sin(4t)$, stationary covariance function $s^2\text{Matern}_{\text{cor}}(d; \rho, \nu)$, and noises $\sim N(0, (s/r)^2)$ by

```
GausFD_cgrid = sim_gfd(pgrid, n, s, r, nu, rho, dense, cgrid, stat);
```

where `cgrid` is a Boolean indicator that controls the output as either common-grid (with `cgrid = 1`) data on `pgrid` or uncommon-grid data with a randomly selected proportion (given by `dense`) of observation grid points (with `cgrid = 0`) from `pgrid`. In addition, `stat` specifies simulating stationary data (with `stat = 1`) from $\mathrm{GP}(3\sin(4t), s^2\mathrm{Matern}_{\mathrm{cor}}(d;\rho,\nu))$, or simulating data from a nonlinearly transformed GP (with `stat = 0`) with mean function $\mu(t) = 3(t+0.5)\sin(4t^{2/3})$ and nonstationary covariance function $\Sigma(t,t') = s^2(t+0.5)(t'+0.5)\mathrm{Matern}_{\mathrm{cor}}(|t^{2/3}-t'^{2/3}|;\rho,\nu)$.

Here, p denotes the length of pooled-grid `pgrid`. The output `GausFD_cgrid` is a data structure consisting of a cell of true data ($\mathrm{Xtrue\_cell}_{1\times n}$), a cell of noisy data ($\mathrm{Xraw\_cell}_{1\times n}$), a cell of observation grids ($\mathrm{Tcell}_{1\times n}$), a true covariance matrix ($\mathrm{Cov\_true}_{p\times p}$), and a true mean vector ($\mathrm{Mean\_true}_{1\times p}$).

3.2. Analyze stationary functional data by BHM

Stationary functional data with common grids

First, we need to setup the required parameter structure with function `setOptions_bfda`. For example, to analyze functional data with common observation grids and stationary covariance function by BHM, the structure `param` can be set as

```
param = setOptions_bfda('smethod', 'bhm', 'cgrid', 1, 'mat', 1, 'M', ...
  10000, 'Burnin', 2000, 'w', 1, 'ws', 1);
```

where each parameter is followed by its value, and unspecified parameters are taken as default values (Appendix A.1). Specifically, `smethod = 'bhm'` denotes using the BHM method; `cgrid = 1` denotes the analyzed data are of common-grid; `mat = 1` denotes taking $A(\cdot,\cdot)$ in Equation 1 as a Matérn correlation function; `M = 10000` denotes the number of MCMC iterations; `Burnin = 2000` denotes the number of MCMC burn-ins; `w = 1` and `ws = 1` are used to tune the Gamma priors for σ_ϵ^2 and σ_s^2.

To analyze the stationary functional data with common grid by BHM, we can then call the main function `BFDA()` by

```
[out_cgrid, param] = BFDA(GausFD_cgrid.Xraw_cell, GausFD_cgrid.Tcell, param);
```

The output structure `out_cgrid` contains smoothed estimates for the signals (`out_cgrid.Z`), mean function (`out_cgrid.mu`), covariance function (`out_cgrid.Sigma`), and other parameters in Equation 1, along with the corresponding 95% point-wise credible intervals (Appendix A.1). The output argument `param` is the updated parameter structure.

Stationary functional data with uncommon grids

To apply BHM on stationary functional data of uncommon-grid, e.g., `GausFD_ucgrid` generated by

```
GausFD_ucgrid = sim_gfd(pgrid, n, s, r, nu, rho, dense, 0, stat);
```

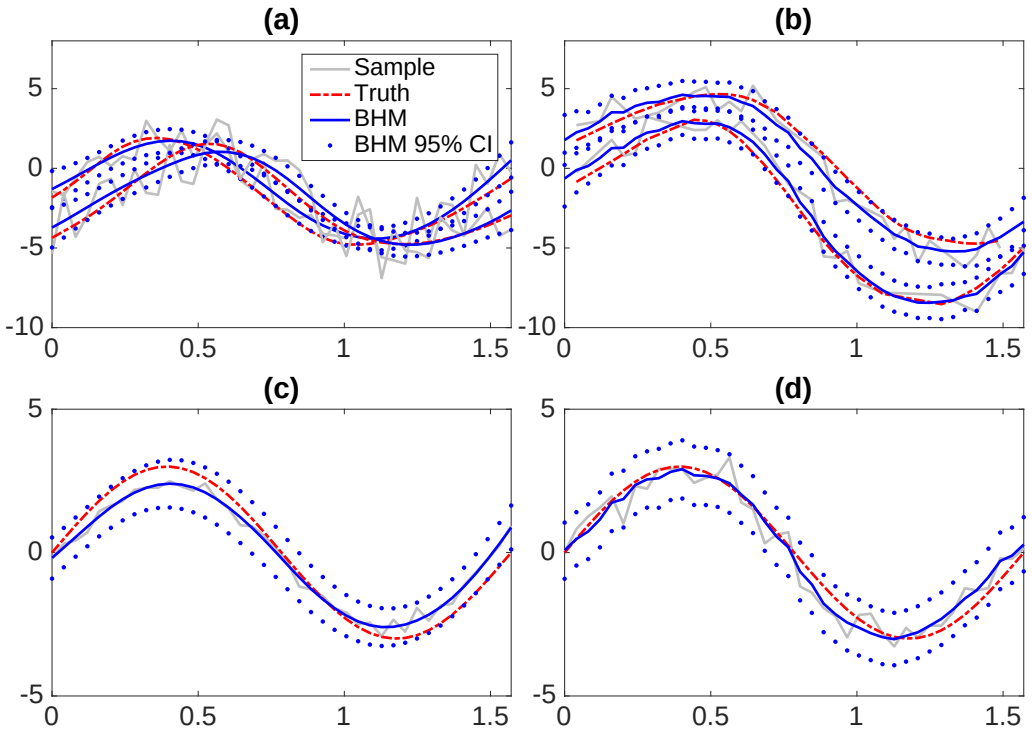

Figure 1: Results of analyzing *stationary* functional data by BHM: (a) two sample signal estimates with common grids; (b) two sample signal estimates with uncommon grids; (c) mean estimate with common grids; (d) mean estimate with uncommon grids; along with 95% point-wise credible intervals (blue dots).

the main function `BFDA` can be called by

```
param_uc = setOptions_bfda('smethod', 'bhm', 'cgrid', 0, 'mat', 1, 'M', ...
  10000, 'Burnin', 2000, 'pace', 1, 'ws', 0.1);
[out_ucgrid, param_uc] = BFDA(GausFD_ucgrid.Xraw_cell, ...
  GausFD_ucgrid.Tcell, param_uc);
```

Here `cgrid` is set as 0 in `param_uc` to specify the functional data input are observed on uncommon grids.

Example BHM results with stationary function data input

In Figure 1(a, b), for both scenarios with common and uncommon grids, we show that the smoothed functional samples by BHM (blue solid) well represent the truth (red dashed), with coverage probabilities by 95% point-wise credible intervals (blue dotted) > 0.95. In addition, the nonparametric mean estimates by BHM (blue solid curves in Figure 1(c, d)) are also smooth and well represent the truth (red dashed),with coverage probabilities > 0.95 by corresponding 95% point-wise credible intervals (blue dotted).

In addition, we show that the Bayesian nonparametric covariance estimates in Figure 2(a, b) are clearly smoother than the sample covariance estimate by using the raw common-grid data in Figure 2(c), and well represent the true stationary covariance as shown in Figure 2(d).

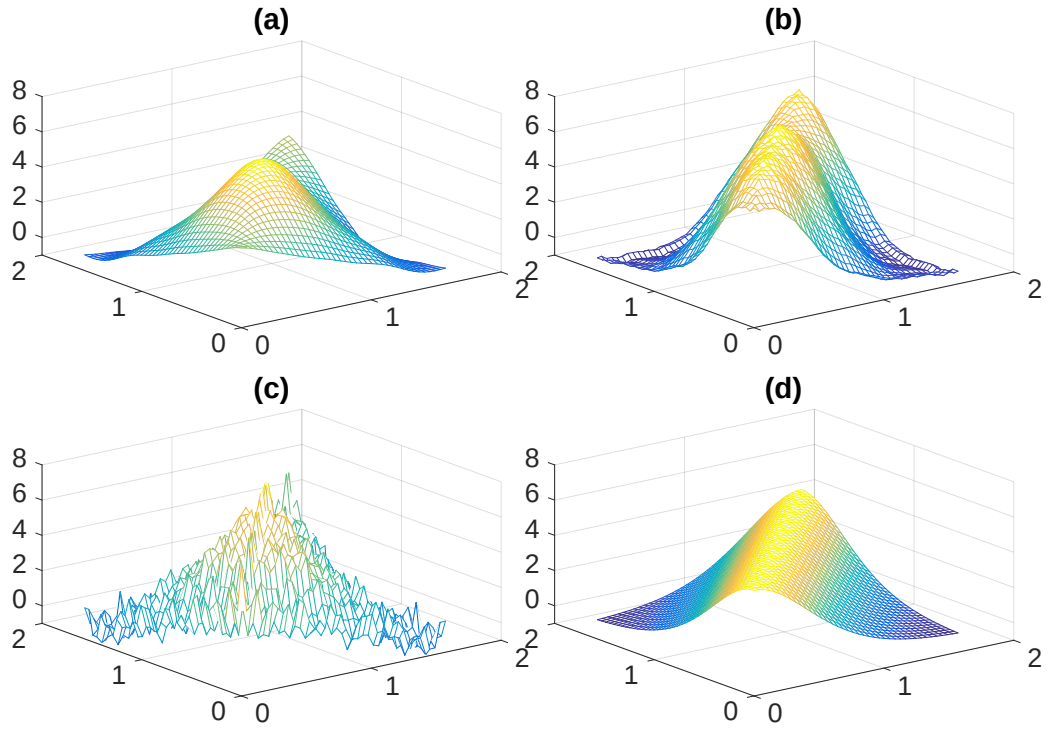

Figure 2: Results of covariance estimation with *stationary* functional data: (a) BHM estimate with common grids; (b) BHM estimate with uncommon grids; (c) sample estimate with raw common-grid data; (d) true covariance surface.

Importantly, although 40% information is lost for the uncommon-grid scenario, BHM still produces similarly good smoothing and estimation results as using completely observed data on pooled grid.

3.3. Analyze nonstationary functional data by BHM

Nonstationary functional data with common grids

To apply BHM on nonstationary functional data with common grids, e.g., functional data (`GausFD_cgrid_ns`) simulated by:

```
GausFD_cgrid_ns = sim_gfd(pgrid, n, s, r, nu, rho, dense, cgrid, 0);
```

the main function `BFDA()` can be called by

```
param_ns = setOptions_bfda('smethod', 'bhm', 'cgrid', 1, 'mat', 0, 'M', ...
  10000, 'Burnin', 2000, 'pace', 1, 'ws', 0.01);
[out_cgrid_ns, param_ns] = BFDA(GausFD_cgrid_ns.Xraw_cell, ...
  GausFD_cgrid_ns.Tcell, param_ns);
```

Here, $A(\cdot,\cdot)$ in Section 2.1 is set as the empirical estimate (`mat = 0`) by **PACE** (Yao *et al.* 2005a, 2015) with `pace = 1` (default), or by the sample covariance estimate using smoothed data by CSS with `pace = 0`.

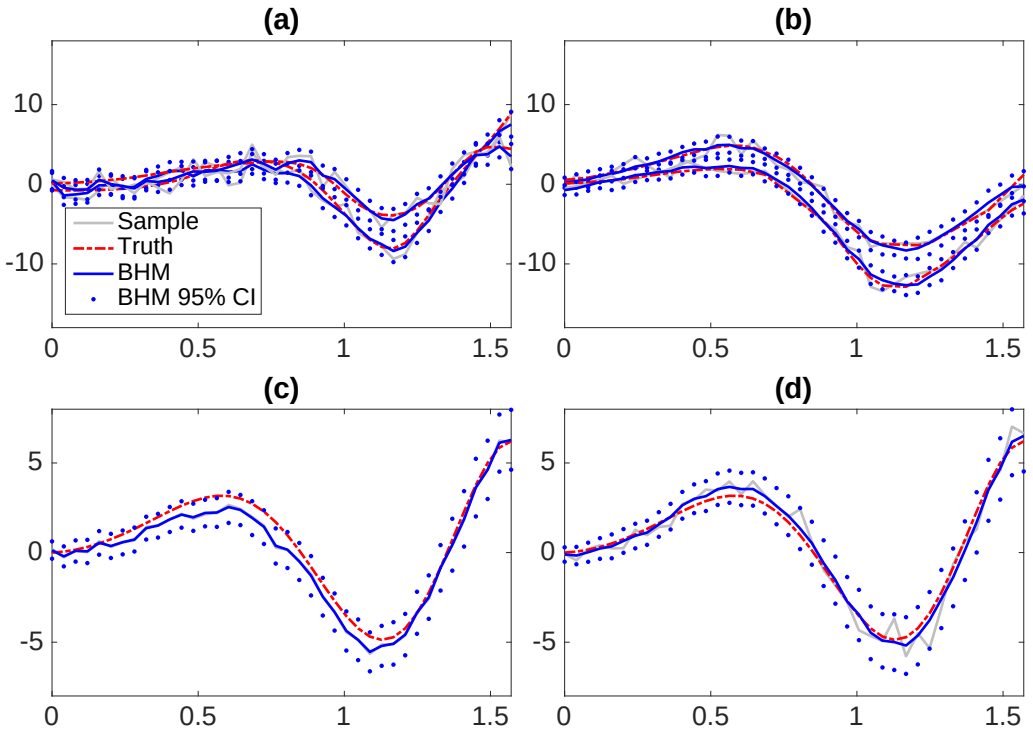

Figure 3: Results of analyzing *nonstationary* functional data by BHM: (a) two sample signal estimates with common grids; (b) two sample signal estimates with uncommon grids; (c) mean estimate with common grids; (d) mean estimate with uncommon grids; along with 95% point-wise credible intervals (blue dots).

Nonstationary functional data with uncommon grids

To analyze nonstationary functional data collected on uncommon (sparse) grids, e.g., functional data (`GausFD_ucgrid_ns`) simulated by

```
GausFD_ucgrid_ns = sim_gfd(pgrid, n, s, r, nu, rho, dense, 0, 0);
```

we can call the main function `BFDA()` by

```
param_uc_ns = setOptions_bfda('smethod', 'bhm', 'cgrid', 0, 'mat', 0, ...
  'M', 10000, 'Burnin', 2000, 'pace', 1, 'ws', 0.01);
[out_ucgrid_ns, param_uc_ns ] = BFDA(GausFD_ucgrid_ns.Xraw_cell, ...
  GausFD_ucgrid_ns.Tcell, param_uc_ns);
```

where variables `cgrid` and `mat` are set as 0.

Example BHM results with nonstationary functional data input

Similarly, as shown in Figures 3 and 4, the BHM estimates of functional signals and mean-covariance functions well represent the truth. Specifically, the 95% point-wise credible intervals of the BHM signal estimates have coverage probabilities > 0.95. Although BHM overestimated the scale of covariance function, BHM captured the major covariance structure

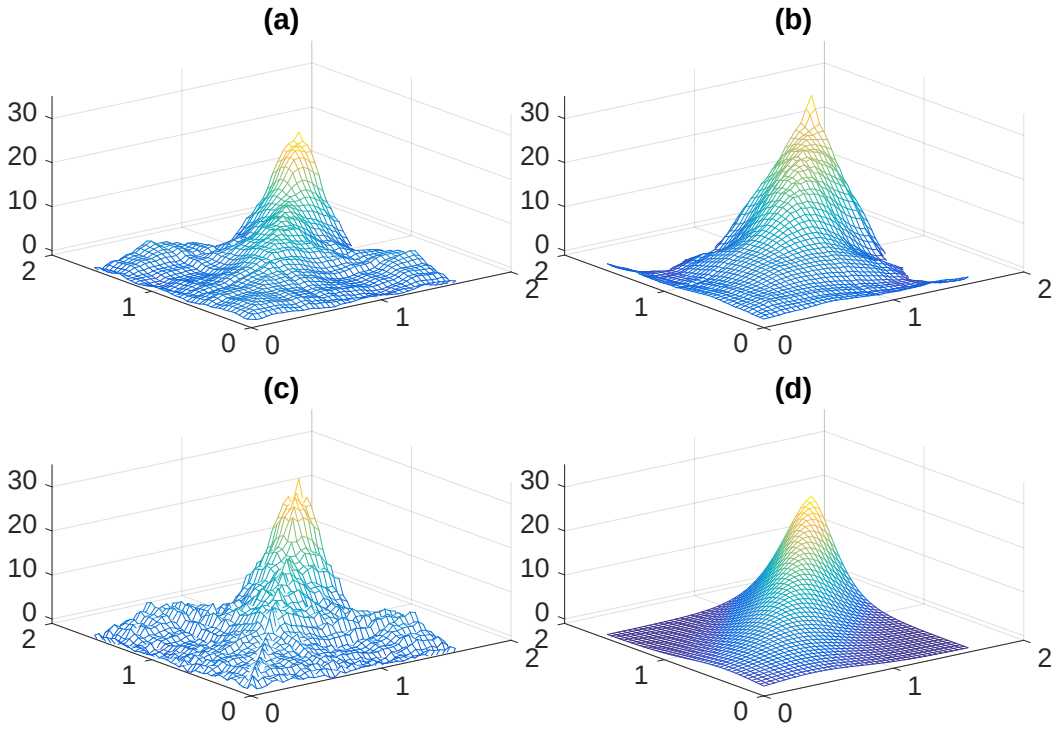

Figure 4: Results of covariance estimates for *nonstationary* functional data: (a) BHM estimate with common grids; (b) BHM estimate with uncommon grids; (c) sample estimate with raw common-grid data; (d) true covariance surface.

and produced a smoothed estimate. The magnitude of the covariance estimate by BHM can be tuned by `ws`, where a smaller `ws` will relatively shrink the magnitude of BHM covariance estimate. We suggest users to tune this parameter according to the magnitude of sample covariance estimate.

3.4. Analyze functional data with random observation grids by BABF

To analyze functional data with random (or high dimensional) observation grids, we recommend users to use the efficient BABF inference algorithm. In addition, **BFDA** also provides the convenience to simulate stationary and nonstationary functional data with random observation grids from the same GPs as in Section 3.1. For example, a structure of functional data `GausFD_rgrid`, with `n` independent functional samples and `p` random observation points per sample (uniformly generated from the observation domain [`au`, `bu`]), can be generated by

```
GausFD_rgrid = sim_gfd_rgrid(n, p, au, bu, s, r, nu, rho, stat);
```

Here, `stat` specifies simulating from the stationary (`stat = 1`) GP or simulating from the nonstationary (`stat = 0`) GP.

Stationary functional data with random grids

To analyze stationary functional data by BABF, simply call the main function `BFDA()` by:

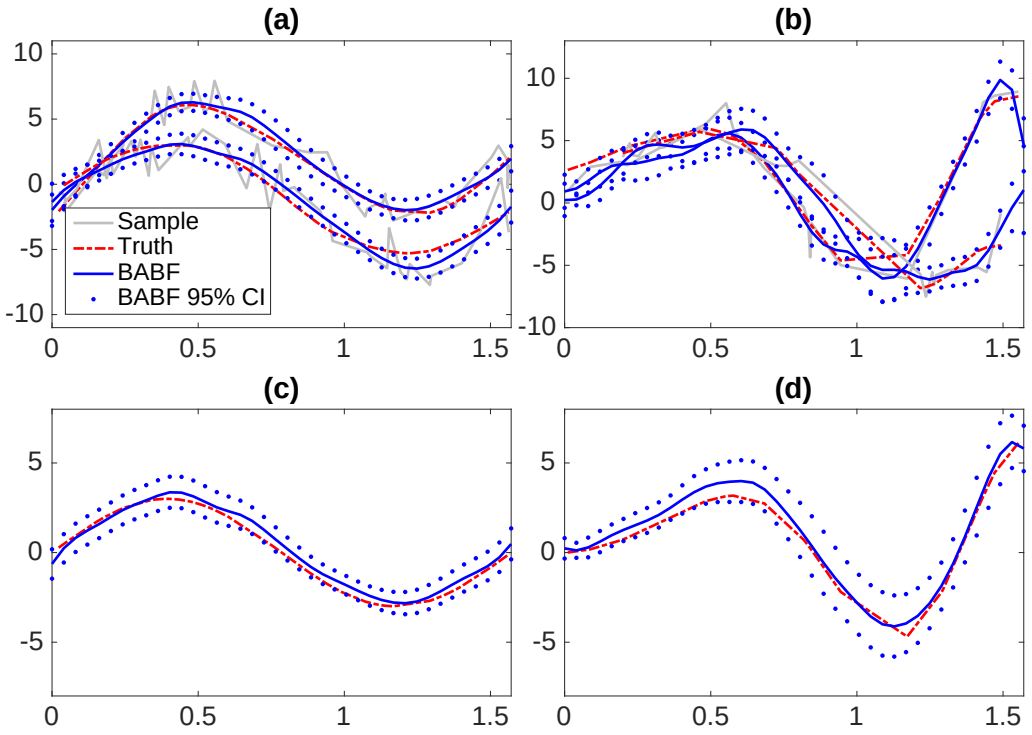

Figure 5: Results of analyzing functional data with *random grids* by BABF: (a) two sample signal estimates with stationary data; (b) two sample signal estimates with nonstationary data; (c) mean estimate with stationary data; (d) mean estimate with nonstationary data; along with 95% point-wise credible intervals (blue dots).

```
param_rgrid = setOptions_bfda('smethod', 'babf', 'cgrid', 0, 'mat', 1, ...
  'M', 10000, 'Burnin', 2000, 'm', m, 'eval_grid', pgrid, 'ws', 1);
[out_rgrid, param_rgrid]= BFDA(GausFD_rgrid.Xraw_cell, ...
  GausFD_rgrid.Tcell, param_rgrid);
```

Here, the working grid $\boldsymbol{\tau}$ can be either set up with user-specified grid in `param_rgrid`, or set as the equally spaced quantiles of the pooled grid by default with user-specified dimension `m`.

Nonstationary functional data with random grids

To implement BABF with nonstationary functional data such as `GausFD_rgrid_ns` simulated by

```
GausFD_rgrid_ns = sim_gfd_rgrid(n, p, au, bu, s, r, nu, rho, 0);
```

we can call the main function `BFDA()` by

```
param_rgrid_ns = setOptions_bfda('smethod', 'babf', 'cgrid', 0, 'mat', 0, ...
  'M', 10000, 'Burnin', 2000, 'm', m, 'eval_grid', pgrid, 'ws', 0.05);
[out_rgrid_ns, param_rgrid_ns] = BFDA(GausFD_rgrid_ns.Xraw_cell, ...
  GausFD_rgrid_ns.Tcell, param_rgrid_ns);
```

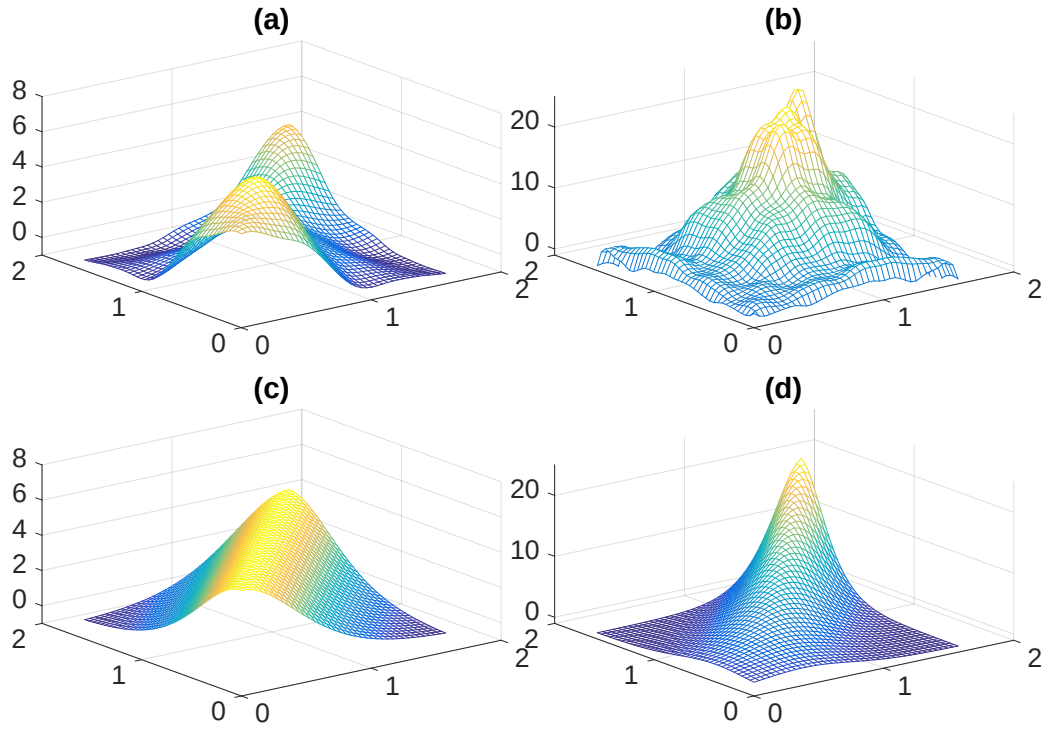

Figure 6: Results of covariance estimation for functional data with *random grids*: (a) BABF estimate with stationary data; (b) BABF estimate with nonstationary data; (c) true covariance surface for stationary data; (d) true covariance surface for nonstationary data.

where `mat` is set as 0 in `param_rgrid_ns`.

Example BABF results with random observation grids

BABF method can efficiently analyze both stationary and nonstationary functional data with random observation grids, producing smooth estimates for functional signals and mean-covariance functions that well represent the truth (Figures 5 and 6). Particularly, the 95% point-wise credible intervals of signal estimates have coverage probabilities > 0.95.

4. Functional linear regression with smoothed data

We expect that follow-up FDA results will be improved by using the accurately smoothed functional data produced by **BFDA** (especially by BHM and BABF). Here, we show example results of functional linear regression under the following two models (with scalar and functional responses respectively),

$$\boldsymbol{Y} = \beta_0 + \int \boldsymbol{X}(t)^\top \boldsymbol{\beta}(t)\, dt + \boldsymbol{\epsilon}, \tag{4}$$

$$\boldsymbol{Y}(t) = \beta_0(t) + \boldsymbol{X}(t)^\top \boldsymbol{\beta}(t) + \boldsymbol{\epsilon}(t), \tag{5}$$

where

- $\boldsymbol{Y}$ in Equation 4 denotes a $n \times 1$ vector of scalar responses; $\boldsymbol{Y}(t) = (y_1(t), \ldots, y_n(t))^\top$ in Equation 5 denotes a vector of functional responses;
- $\boldsymbol{X}(t)$ denotes a $n \times q$ design matrix of q functional independent variables;
- $\boldsymbol{\beta}(t)$ denotes a $q \times 1$ vector of coefficient functions for independent variables;
- β_0 and $\beta_0(t)$ denote the intercept terms;
- $\boldsymbol{\epsilon}$ and $\boldsymbol{\epsilon}(t)$ denote the error terms.

Note that $\boldsymbol{X}(t)$ and $\boldsymbol{\beta}(t)$ can also denote nonfunctional covariates and coefficients, because nonfunctional variables are basically constant functions of t.

4.1. Simulate functional data

To evaluate the improvement of regression results using smoothed data by the BABF method implemented in **BFDA**, we first simulated 30 raw stationary GP trajectories $\{X_i(\boldsymbol{t}_i)\}$ with random grids from domain $[0, \pi/2]$, which were generated by `sim_gfd_rgrid(30, 40, 0, 1.5708, 2.2361, 2, 3.5, 0.5, 1)`. Then we simulated scalar responses by

$$Y_i = \int_0^{1.5708} X_i(t)t^2\, dt + \epsilon,\ \epsilon \sim N(0, 1);$$

and functional responses by

$$Y_i(t) = X_i(t)t^2 + \epsilon,\ \epsilon \sim N(0, 1).$$

Because functional regression function `fRegress()` from **fdaM** package requires inputs of functional data with common grids, we used the function `csaps()` with the suggested optimal smoothing parameter 1 (CSS approach) to interpolate true functional data, smoothed functional data by BABF, and noisy functional data on the same equally spaced grid (with length 40) over $[0, \pi/2]$. The interpolated functional data from raw functional data are equivalent to the individually smoothed ones by CSS (one example curve is shown in Figure 7(a)).

Using the smoothed data by BABF and CSS, we respectively fitted the functional linear models (Equations 4 and 5) using 20 randomly chosen signals, and then tested the prediction results using the remains. We replicated this fitting process for 100 times, and evaluated the performance by the average mean square errors (MSEs) of the fitted and predicted responses.

4.2. Results of functional linear regression with scalar responses

For the scenario with scalar responses, although the fitted coefficient functions using both smoothed data by BABF and CSS are close to the truth (Figure 7(b, c)), with coverage probabilities > 0.95 for the 95% confidence intervals, the average MSEs of the fitted and predicted responses from 100 replications are smaller for using the BABF smoothed data than the ones using the CSS smoothed data (0.311 vs. 0.388 for fitted responses, 0.497 vs. 0.677 for predicted responses, as shown in Table 1). Figure 8 shows the results of an example replication of this fitting and predicting process with scalar responses.

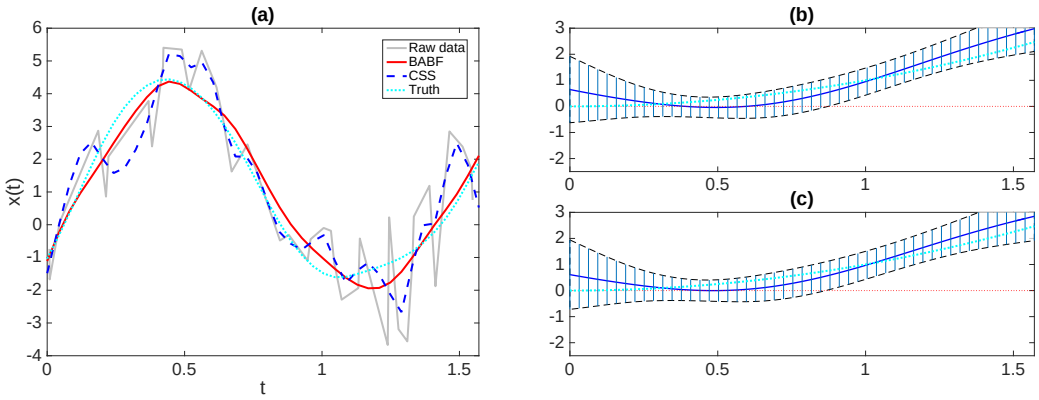

Figure 7: (a) Example estimate of $X_i(t)$; (b) the estimate of $\beta(t)$ using the smoothed data by BABF; (c) the estimate of $\beta(t)$ using the smoothed data by CSS along with the truth in the cyan dotted lines.

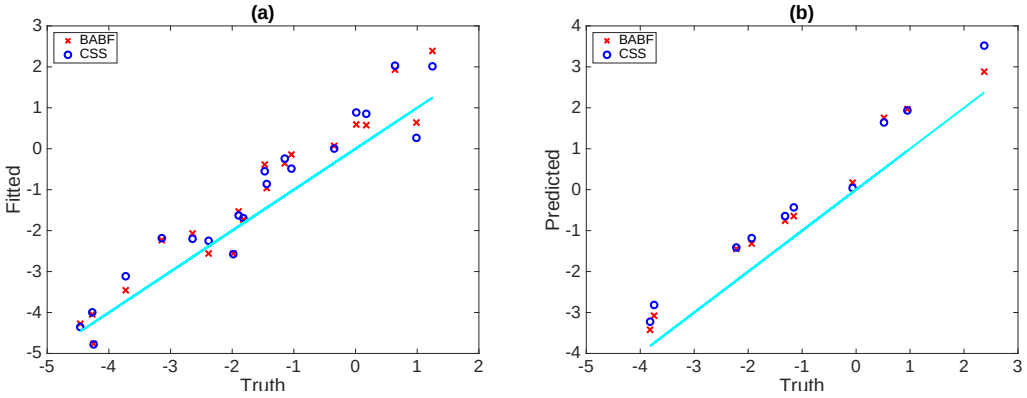

Figure 8: (a) Fitted vs. true scalar responses; (b) predicted vs. true scalar responses.

4.3. Results of functional linear regression with functional responses

For the scenario with functional responses, we can see that the fitted intercept term using BABF smoothed data is closer to the truth (constant 0) with narrower 95% confidence interval than the one using CSS smoothed data (Figure 9(a, c)). In addition, the coefficient function using BABF smoothed data has narrower 95% confidence interval and higher coverage probability (Figure 9(b, d)). Consequently, both fitted and predicted functional responses using BABF smoothed data have smaller point-wise MSEs out of 100 replications, 0.417 vs. 1.190 for fitted functional responses, 0.464 vs. 1.354 for predicted functional responses (Table 1). Figure 10 shows the results of an example replication of this fitting and predicting process with functional responses.

5. Discussion

The MATLAB tool **BFDA** presented in this paper can simultaneously smooth multiple functional observations and estimate the mean-covariance functions, assuming the functional data

MSE (std)	BABF smoothed		CSS smoothed	
	$\boldsymbol{Y}$	$\boldsymbol{Y}(t)$	$\boldsymbol{Y}$	$\boldsymbol{Y}(t)$
Fitted	0.311 (0.061)	0.417 (0.049)	0.388 (0.074)	1.190 (0.186)
Predicted	0.497 (0.289)	0.464 (0.112)	0.677 (0.435)	1.354 (0.419)

Table 1: Average MSEs of the fitted and predicted responses for 100 replicates, along with the standard deviations of these MSEs among 100 replicates in the parentheses, for scalar responses $\boldsymbol{Y}$ and functional responses $\boldsymbol{Y}(t)$.

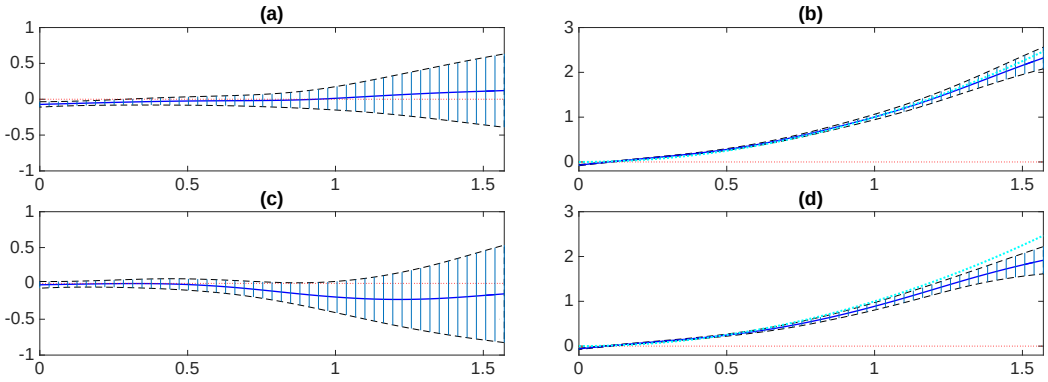

Figure 9: (a) Estimated intercept function $\beta_0(t)$ using BABF smoothed data; (b) Estimated coefficient function $\beta(t)$ using BABF smoothed data; (c) Estimated intercept function $\beta_0(t)$ using CSS smoothed data; (d) Estimated coefficient function $\beta(t)$ using CSS smoothed data; along with 95% confidence intervals and true coefficient functions in cyan dotted lines.

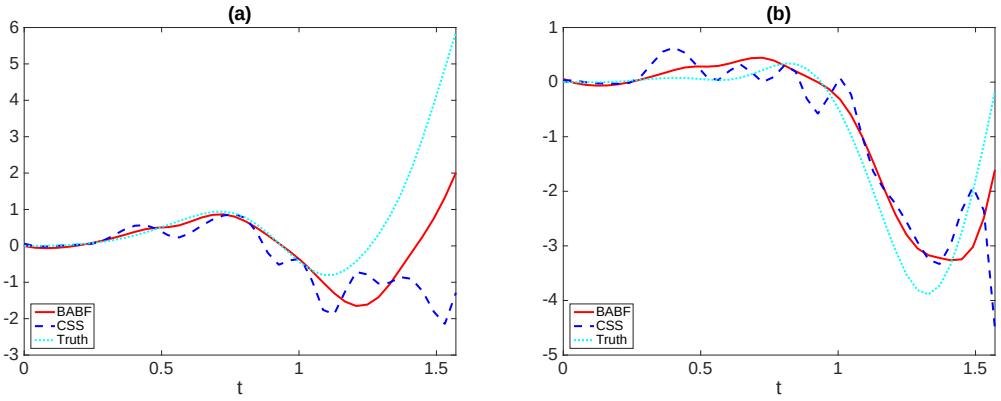

Figure 10: (a) Example fitted functional responses; (b) example predicted functional responses; with true signals in cyan dotted lines.

are of the same GP distribution. The smoothed data by **BFDA** are shown to be more accurate than the conventional individual smoothing methods such as CSS, thus improving follow-up FDA results. The advantages of **BFDA** include:

- Simultaneously smoothing multiple functional samples and estimating mean-covariance functions in a nonparametric way;

- Flexibly handling functional data with stationary and nonstationary covariance functions, common or uncommon (sparse) observation grids;
- Efficiently dealing with high-dimensional functional data or functional data with random observation grids by the efficient BABF algorithm.

BFDA is suitable for analyzing data that can be roughly assumed as from the same GP distribution. We recommend using the BHM method for low-dimensional functional data with common grids or sparse functional data, and using the BABF method for high-dimensional functional data with dense grids (including both common and random grids). In addition, we recommend using the Matérn function as the prior covariance structure for analyzing functional data with stationary covariance functions, while using the empirical covariance estimate (e.g., the estimate by **PACE**) for analyzing functional data with nonstationary covariance functions.

The follow-up functional data analysis can be conducted using existing software packages (e.g., **fdaM** in MATLAB, Ramsay 2014; **fda** in R, Ramsay *et al.* 2018). Examples are provided in the supplementary file about using the **fdaM** package with functional data smoothed by **BFDA**. Details about the inputs and outputs of **BFDA** are provided in Appendix A. The most recent version of **BFDA** tool and example scripts are freely available from `https://github.com/yanglab-emory/BFDA`.

References

Crainiceanu CM, Goldsmith AJ (2010). "Bayesian Functional Data Analysis Using **WinBUGS**." *Journal of Statistical Software*, **32**(11), 1–33. doi:10.18637/jss.v032.i11.

Dawid AP (1981). "Some Matrix-Variate Distribution Theory: Notational Considerations and a Bayesian Application." *Biometrika*, **68**(1), 265–274. doi:10.1093/biomet/68.1.265.

De Boor C (1977). "Computational Aspects of Optimal Recovery." In CA Micchelli, TJ Rivlin (eds.), *Optimal Estimation in Approximation Theory*, pp. 69–91. Springer-Verlag, Boston. doi:10.1007/978-1-4684-2388-4_3.

Febrero-Bande M, de la Fuente M (2012). "Statistical Computing in Functional Data Analysis: The R Package **fda.usc**." *Journal of Statistical Software*, **51**(4), 1–28. doi:10.18637/jss.v051.i04.

Gaffney PW, Powell MJD (1976). *Optimal Interpolation*. Springer-Verlag.

Gelman A, Rubin DB (1992). "Inference from Iterative Simulation Using Multiple Sequences." *Statistical Science*, **7**(4), 457–472. doi:10.1214/ss/1177011136.

Geman S, Geman D (1984). "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **6**, 721–741. doi:10.1109/tpami.1984.4767596.

Graves S, Hooker G, Ramsay J (2010). *Functional Data Analysis with R and MATLAB*. Springer-Verlag. doi:10.1007/978-0-387-98185-7.

Green PJ, Silverman BW (1993). *Nonparametric Regression and Generalized Linear Models: A Roughness Penalty Approach*. CRC Press. doi:10.1201/b15710.

Hunyadi L (2010). ***bspline**: Draw, Manipulate and Reconstruct B-Splines*. MATLAB package, URL http://www.mathworks.com/matlabcentral/fileexchange/27374-b-splines.

James M (1978). "The Generalised Inverse." *The Mathematical Gazette*, **62**(420), 109–114. doi:10.2307/3617665.

Micchelli CA, Rivlin TJ, Winograd S (1976). "The Optimal Recovery of Amooth Functions." *Numerische Mathematik*, **26**(2), 191–200. doi:10.1007/bf01395972.

Ramsay JO (2014). ***fdaM**: Functional Data Analysis*. MATLAB package, URL http://www.psych.mcgill.ca/misc/fda/downloads/FDAfuns/Matlab/.

Ramsay JO, Dalzell CJ (1991). "Some Tools for Functional Data Analysis." *Journal of the Royal Statistical Society B*, **53**(3), 539–572. doi:10.2307/2981865.

Ramsay JO, Silverman BW (2002). *Applied Functional Data Analysis: Methods and Case Studies*, volume 77. Springer-Verlag. doi:10.1007/b98886.

Ramsay JO, Silverman BW (2005). *Functional Data Analysis*. Springer Series in Statistics, 2nd edition. Springer-Verlag. doi:10.1007/978-1-4757-7107-7.

Ramsay JO, Wickham H, Graves S, Hooker G (2018). ***fda**: Functional Data Analysis*. R package version 2.4.8, URL https://CRAN.R-project.org/package=fda.

Rasmussen CE, Williams CKI (2006). *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge.

R Core Team (2019). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Särkkä S, Aki V (2014). "MCMC Diagnostics for MATLAB." URL http://becs.aalto.fi/en/research/bayes/mcmcdiag/.

Shi JQ, Cheng Y (2014). ***GPFDA**: Apply Gaussian Process in Functional Data Analysis*. R package version 2.2, URL https://CRAN.R-project.org/package=GPFDA.

Shi JQ, Choi T (2011). *Gaussian Process Regression Analysis for Functional Data*. CRC Press, Boca Raton, FL.

Shi JQ, Wang B, Murray-Smith R, Titterington DM (2007). "Gaussian Process Functional Regression Modeling for Batch Data." *Biometrics*, **63**(3), 714–723. doi:10.1111/j.1541-0420.2007.00758.x.

Sturtz S, Ligges U, Gelman A (2005). "**R2WinBUGS**: A Package for Running **WinBUGS** from R." *Journal of Statistical Software*, **12**(3), 1–16. doi:10.18637/jss.v012.i03.

The MathWorks Inc (2017). *MATLAB – The Language of Technical Computing, Version R2017b*. Natick. URL http://www.mathworks.com/products/matlab/.

Vieu P, Ferraty F (2006). *Nonparametric Functional Data Analysis: Theory and Practice*. Springer-Verlag.

Yang J, Cox DD, Lee JS, Ren P, Choi T (2017). "Efficient Bayesian Hierarchical Functional Data Analysis with Basis Function Approximations Using Gaussian-Wishart Processes." *Biometrics*, **73**(4), 1082–1091. doi:10.1111/biom.12705.

Yang J, Zhu H, Choi T, Cox DD (2016). "Smoothing and Mean-Covariance Estimation of Functional Data with a Bayesian Hierarchical Model." *Bayesian Analysis*, **11**(3), 649–670. doi:10.1214/15-ba967.

Yao F, Müller HG, Wang JL (2005a). "Functional Data Analysis for Sparse Longitudinal Data." *Journal of the American Statistical Association*, **100**(470), 577–590. doi:10.1198/016214504000001745.

Yao F, Müller HG, Wang JL (2005b). "Functional Linear Regression Analysis for Longitudinal Data." *The Annals of Statistics*, **33**(6), 2873–2903. doi:10.1214/009053605000000660.

Yao F, Müller HG, Wang JL (2015). ***PACE*** *Package for Functional Data Analysis and Empirical Dynamics (MATLAB)*. Version 2.17, URL http://www.stat.ucdavis.edu/PACE/.

Yuan Y, Johnson VE (2012). "Goodness-of-Fit Diagnostics for Bayesian Hierarchical Models." *Biometrics*, **68**(1), 156–164. doi:10.1111/j.1541-0420.2011.01668.x.

A. Inputs and outputs

A.1. Input variables

The main function `BFDA()` has three input arguments:

- A cell containing all functional data.
- A cell containing all grids on which functional data are observed.
- A parameter structure outputted by function `setOptions_bfda()`, containing all required parameters:
  - `smethod`, specifying the method used for analyzing the functional data. Default value is `'babf'` for BABF method with basis function approximation; other choices are `'bhm'` for BHM method without basis function approximation, `'bgp'` for standard Bayesian GP regression, `'bfpca'` for Bayesian principal components analysis, and `'css'` for cubic smoothing spline.
  - `Burnin`, the number of burn-ins for the MCMC algorithm. Default value is `2000`.
  - `M`, the number of iterations for the MCMC algorithm. Default value is `10000`.
  - `cgrid`, set as `1` if the functional data are observed on a common-grid, otherwise set as `0` for uncommon or random grids. Default value is `1`.
  - `Sigma_est`, estimated smooth covariance matrix from previous analysis. Default is empty and will be estimated by **PACE** or sample estimate from individually smoothed data.
  - `mu_est`, estimated smooth functional mean from previous analysis. Default is empty and will be set as the smoothed sample mean.
  - `mat`, set as `1` to use the Mateŕn covariance function as prior structure for stationary functional data; set as `0` to use the empirical covariance estimate `Sigma_est` as the prior structure for nonstationary functional data. Default value is `1`.
  - `nu`, order of smoothness for the Mateŕn covariance function. Default is empty and will be estimated based on `Sigma_est`.
  - `delta`, shape parameter δ of the IWP. Default is `5` for a non-informative prior.
  - `c`, determining the prior covariance for functional mean. Default is `1`.
  - `w, ws`, determining the prior gamma distributions for σ_ϵ^2 and σ_s^2. Defaults are `w = 1`, `ws = 0.1`. The parameter `ws` should be tuned for a proper magnitude of the posterior covariance estimate.
  - `pace`, if `Sigma_est` and `mu_est` are empty, set `pace = 1` to obtain `Sigma_est` and `mu_est` by **PACE**, and set `pace = 0` to use the empirical estimates from the individually smoothed data by CSS. Default is `1`.
  - `m, tau`, working grid `tau` is only required for `'babf'` method. Default is empty and will be set up as the $(0 : \frac{100}{m-1} : 100)$ percentiles of the pooled observation grid with length `m`.
  - `eval_grid`, evaluation grid for all functional estimates, only required for `'babf'` methods.

  - `lamb_min, lamb_max, lamb_step`, determining the smoothing parameter candidates for general cross validation of the CSS method. Defaults are `lamb_min = 0.9`, `lamb_max = 0.99`, `lamb_step = 0.01`.
  - `a, b`, hyper-parameters for the gamma distributions in `'bgp'`, and `'bfpca'`.
  - `resid_thin`, determine the MCMC thinning steps of the residuals that are used to test the goodness-of-fit of the model. Default is `10`.

A.2. Output variables

The main function `BFDA()` has two output arguments, one structure outputted by the specified method, and the other the parameter structure as specified by `setOptions_bfda()` containing updated parameter values.

Output structure with `smethod = 'bhm'`:

- `Z, Z_CL, Z_UL`, smoothed functional data, lower and upper 95% credible intervals.
- `Sigma, Sigma_CL, Sigma_UL`, functional covariance estimate, lower and upper 95% credible intervals.
- `Sigma_SE`, the empirical covariance estimate by using the smoothed data `Z`.
- `mu, mu_CI`, functional mean estimate, 95% credible intervals.
- `rn, rn_CI`, estimate and 95% credible interval for the noise precision.
- `rs, rs_CI`, estimate and 95% credible interval for σ_s^2.
- `rho, nu`, estimated parameter values for the Matérn function.
- `residuals`, MCMC samples of the residuals that are used to test the goodness-of-fit.
- `pmin_vec`, p values for testing the goodness-of-fit for all functional samples. A p value > 0.25 suggests no evidence of model inadequacy; $0.05 < p$ value < 0.25 suggests some evidence of model inadequacy; p value < 0.05 suggests strong evidence of model inadequacy.

The output structure with `smethod = 'babf'` has the following variables that are different from the ones with `smethod = 'bhm'`:

- `Zt`, smoothed functional data on the observation grids.
- `Z_cgrid, Z_cgrid_CL, Z_cgrid_UL`, smoothed functional data on the evaluation grid `eval_grid`, along with lower and upper 95% credible intervals.
- `Sigma_cgrid, Sigma_cgrid_CL, Sigma_cgrid_UL`, functional covariance estimate on the evaluation grid `eval_grid`, along with lower and upper 95% credible intervals.
- `mu_cgrid, mu_cgrid_CI`, functional mean estimate on the evaluation grid `eval_grid`, along with 95% credible intervals.

- `Zeta, Zeta_CL, Zeta_UL`, estimates for the coefficients of basis functions, along with lower and upper 95% credible intervals.
- `Sigma_zeta_SE`, the empirical covariance estimate with the estimated `Zeta`.
- `Sigma_zeta, Sigma_zeta_CL, Sigma_zeta_UL`, covariance estimate for the coefficients of basis functions, along with lower and upper 95% credible intervals.
- `mu_zeta, mu_zeta_CI`, mean estimate for the coefficients of basis functions, along with 95% credible intervals.
- `Btau`, the basis function evaluations on the working grid `tau`.
- `BT`, the basis function evaluations on the observation grids.
- `Sigma_tau`, functional covariance estimate on the working grid `tau`.
- `mu_tau`, functional mean estimate on the working grid `tau`.
- `optknots`, the optimal knots selected by `optknt()` for evaluations on the working grid `tau`.

Affiliation:

Jingjing Yang, Ph.D.
Center for Computational and Quantitative Genetics
Department of Human Genetics
Emory University School of Medicine
615 Michael St.
Atlanta, GA, 30322, United States of America
E-mail: jingjing.yang@emory.edu, yjingj@gmail.com

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
May 2019, Volume 89, Issue 2 *Submitted:* 2016-04-16
doi:10.18637/jss.v089.i02 *Accepted:* 2017-10-29