

TUTORIAL IN BIOSTATISTICS OPEN ACCESS

# Tutorial on Bayesian Functional Regression Using Stan

Ziren Jiang<sup>1</sup>  | Ciprian Crainiceanu<sup>2</sup> | Erjia Cui<sup>1</sup><sup>1</sup>Division of Biostatistics and Health Data Science, University of Minnesota, Minneapolis, Minnesota, USA | <sup>2</sup>Department of Biostatistics, Johns Hopkins University, Baltimore, Maryland, USA**Correspondence:** Erjia Cui ([ecui@umn.edu](mailto:ecui@umn.edu))**Received:** 10 March 2025 | **Revised:** 26 August 2025 | **Accepted:** 1 September 2025**Funding:** This work was partially supported by the NIH grants R01 NS060910 (CC) and R01 AG075883 (CC).**Keywords:** Bayesian data analysis | functional Cox regression | functional data analysis | functional principal component analysis | stan

## ABSTRACT

This manuscript provides step-by-step instructions for implementing Bayesian functional regression models using Stan. Extensive simulations indicate that the inferential performance of the methods is comparable to that of state-of-the-art frequentist approaches. However, Bayesian approaches allow for more flexible modeling and provide an alternative when frequentist methods are not available or may require additional development. Methods and software are illustrated using the accelerometry data from the National Health and Nutrition Examination Survey (NHANES).

## 1 | Introduction

The collection of ever more complex and high-dimensional data in many scientific areas has led to increased interest in Functional Data Analysis (FDA)—a branch of statistics that focuses on the analysis of data that can be represented as functions, curves, or trajectories observed over a continuum, such as time or space [1–3]. Some examples include continuous glucose monitoring [4], colon carcinogenesis [5], electroencephalography [6], epidemiological monitoring [7], fiber photometry [8], genomics [9], house pricing [10], neuroimaging [11–15], online auction [16], polysomnography [17], traffic flow [18], and wearable and implantable devices [19, 20], among others.

The complexity, size, and structure of the new data sets require the development of new functional analytic methods and, especially, software that can be used, adapted, and tested in reasonable time. In a recent monograph, Crainiceanu et al. [3] showed that functional regression models can be viewed as mixed effects models and existing frequentist software, such as the `refund` [21] and `mgcv` [22] packages in R [23], can be used to fit functional regression models.

Bayesian analysis naturally accommodates hierarchical structures, allows for the incorporation of prior information, and provides a coherent framework for quantifying uncertainty in both fixed and random effects, making it especially well suited for mixed effect models. This suggests that Bayesian analyses may provide a powerful alternative to frequentist approaches that could be adapted faster to emerging data structures. In this paper, we provide a tutorial on scalar-on-function regression (SoFR, where the outcome is scalar and predictors include functional variables), function-on-scalar regression (FoSR, where the outcome is a functional variable and predictors are scalar), and functional Cox regression (for time-to-event outcomes with functional predictors), all using Bayesian methods implemented in Stan [24]. This type of models has been introduced by multiple researchers using different nomenclatures [1, 25]; the first use of the SoFR/FoSR nomenclature can be traced to Reiss et al. [26]. The first Bayesian implementation of these models was provided by Crainiceanu and Goldsmith [27] using WinBUGS [28]. The current tutorial will substantially expand existing Bayesian functional regression software by taking advantage of the rapid development of Bayesian computation in the past 15 years. In particular, we: (1) provide a detailed description of Bayesian

This is an open access article under the terms of the [Creative Commons Attribution](https://creativecommons.org/licenses/by/4.0/) License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

© 2025 The Author(s). *Statistics in Medicine* published by John Wiley & Sons Ltd.

functional regression models and the corresponding `Stan` implementation with step-by-step code demonstrations; (2) propose a Bayesian joint model to incorporate functional principal component analysis (FPCA) into functional regression models and estimate the functional coefficients and principal component scores simultaneously; (3) conduct extensive simulation experiments to validate the software and compare it with the state-of-the-art frequentist software; and (4) provide an R package, `refund-Bayes`, for model implementation. The supplementary R Markdown file contains the complete code based on our `refund-Bayes` package. Some familiarity with splines, regression, and penalized approaches is assumed; see Crainiceanu et al. [3] for an introduction, especially chapters 4–6.

The rest of the manuscript is organized following the logic of functional regression. In Section 2, we introduce the Bayesian scalar-on-function regression model and its implementation in `Stan`. In Section 3, we introduce the Bayesian functional Cox regression model for time-to-event outcome. Both Sections 2 and 3 use the observed functional covariate as the predictor variable. In Section 4, we illustrate how to incorporate FPCA into our Bayesian functional regression model. In Section 5, we introduce the Bayesian function-on-scalar regression model. Section 6 provides real-world application examples using the NHANES data. We conclude with discussions in Section 7.

## 2 | Bayesian Scalar-on-Function Regression

Penalized splines have become the practical standard in semiparametric regression [22, 29–32] because they provide an excellent balance between computational complexity, adaptation to real data scenarios, and inferential capabilities. Penalized splines use a moderately large number of basis functions to account for the maximum complexity of the model and quadratic penalties to control the smoothness of the fit. By showing that these penalized models are equivalent to mixed effects models, software that was originally developed for inference in mixed effects models can be expanded to semiparametric regression. These ideas have been extended to functional regression that also have a mixed effects representation [11, 27]. In this section, we present the `Stan` program for implementing a Bayesian Scalar-on-Function Regression (SoFR) model, where functional parameters are modeled nonparametrically using penalized splines. The inferential performance of these models compared with existing frequentist models are assessed via simulations.

### 2.1 | The SoFR Model

We start by introducing the data structure for SoFR. For subject  $i = 1, \dots, n$ , let  $Y_i$  be the outcome,  $\mathbf{Z}_i$  be the  $p \times 1$  dimensional vector of scalar predictors, and  $\{W_i(t_{im}), t_{im} \in [0, 1]\}$  with  $m = 1, \dots, M_i$  be a functional predictor, where  $M_i$  is the number of functional observations for study participant  $i$ . Here, we assume that the observation time points are identical for all subjects, such that  $M_1 = M_2 = \dots = M_n = M$  and  $t_{im} = t_m$  for all  $i = 1, \dots, n$  and  $m = 1, \dots, M_i$ . For data observed at irregular locations, users can refer to Section 4, where we introduce the Bayesian joint modeling of functional principal component analysis (FPCA) and functional regression. Although we use a single functional predictor

here to illustrate the model, the proposed method and software can be easily extended to multiple functional predictors.

The SoFR model assumes that the distribution of  $Y_i$  follows an exponential family with mean  $\mu_i$ , and the linear predictor  $\eta_i = g(\mu_i)$  has the following structure

$$\eta_i = \eta_0 + \int_0^1 W_i(t)\beta(t)dt + \mathbf{Z}_i'\boldsymbol{\gamma} \quad (1)$$

where  $\eta_0$  is the overall intercept,  $\beta(\cdot) \in L_2[0, 1]$  is the functional coefficient, and  $\boldsymbol{\gamma}$  is a  $p \times 1$  dimensional vector of parameters. If  $\psi_1(t), \dots, \psi_K(t)$  is a collection of  $K$  pre-specified basis functions and  $\beta(t) = \sum_{k=1}^K b_k \psi_k(t)$ , the linear predictor can be re-written as:

$$\begin{aligned} \eta_i &= \eta_0 + \sum_{k=1}^K b_k \int_0^1 W_i(t)\psi_k(t)dt + \mathbf{Z}_i'\boldsymbol{\gamma} \\ &\approx \eta_0 + \sum_{k=1}^K b_k \sum_{m=1}^M L_m W_i(t_m)\psi_k(t_m) + \mathbf{Z}_i'\boldsymbol{\gamma} \\ &= \eta_0 + \mathbf{X}_i'\mathbf{b} + \mathbf{Z}_i'\boldsymbol{\gamma} \end{aligned} \quad (2)$$

where  $L_m = t_{m+1} - t_m$ , and the approximation sign on the second line indicates the Riemann sum approximation to the integral. The  $K \times 1$  dimensional vector  $\mathbf{X}_i = (X_{i1}, \dots, X_{iK})'$  has the  $k$ -th entry equal to

$$X_{ik} = \sum_{m=1}^M L_m W_i(t_m)\psi_k(t_m) \quad (3)$$

Let  $\mathbf{X} = [\mathbf{X}_1, \dots, \mathbf{X}_n]$  be the  $K \times n$  matrix of functional covariates, and  $\mathbf{Z} = [\mathbf{Z}_1, \dots, \mathbf{Z}_n]$  be the  $p \times n$  matrix of scalar covariates. The linear predictor  $\boldsymbol{\eta} = (\eta_1, \dots, \eta_n)'$  can then be expressed as

$$\boldsymbol{\eta} = \eta_0 \mathbf{J}_n + \mathbf{X}'\mathbf{b} + \mathbf{Z}'\boldsymbol{\gamma} \quad (4)$$

where  $\mathbf{J}_n$  is the  $n \times 1$  vector with all entries equal to 1. Note that this, unpenalized, model is a standard generalized linear model (GLM). Smoothness can be induced on the functional regression coefficient,  $\beta(t)$ , by penalizing the spline coefficients,  $\mathbf{b}$ . In the next section we show that this is equivalent to assuming a particular prior on  $\mathbf{b}$ , which transforms the model into a specific generalized linear mixed effects model (GLMM).

### 2.2 | Incorporating Penalized Splines

A common approach to induce smoothness on  $\beta(t)$  is to assume that the number of spline basis functions,  $K$ , is relatively large and add a quadratic penalty on the regression coefficients. We focus on penalties on the integral of the square of the second derivative,  $\int \{\beta''(t)\}^2 dt$ . This penalty was introduced by Grace Wahba and collaborators [33–35] for smoothing spline regression (as many basis functions as observations) and by Finbarr O'Sullivan for penalized spline regression (B-splines with a smaller number of knots than the number of observations) [30]. Given its historical and practical importance, we would like to coin this foundational concept as the Wahba-O'Sullivan smoothing penalty. While we focus on this penalty here, the methods described can be applied to any other type of quadratic penalty and/or basis functions.

The Wahba-O'Sullivan penalty can be re-written as

$$\begin{aligned} \int \{\beta''(t)\}^2 dt &= \int \left\{ \sum_{k=1}^K b_k \psi_k''(t) \right\}^2 dt \\ &= \int \{ \boldsymbol{\psi}''(\mathbf{t}) \}' \mathbf{b} \{ \boldsymbol{\psi}''(\mathbf{t}) \}' \mathbf{b} dt \\ &= \mathbf{b}' \int \{ \boldsymbol{\psi}''(\mathbf{t}) \} \{ \boldsymbol{\psi}''(\mathbf{t}) \}' dt \mathbf{b} = \mathbf{b}' \mathbf{S} \mathbf{b} \end{aligned} \quad (5)$$

where  $\mathbf{S} = \int \{ \boldsymbol{\psi}''(\mathbf{t}) \} \{ \boldsymbol{\psi}''(\mathbf{t}) \}' dt$  is the penalty matrix and  $\mathbf{b} = (b_1, \dots, b_K)'$  are spline coefficients. A penalized spline regression approach would minimize the criterion

$$-2 \log L(\mathbf{Y}|\mathbf{X}, \mathbf{Z}, \eta_0, \mathbf{b}, \gamma) + \lambda \mathbf{b}' \mathbf{S} \mathbf{b} \quad (6)$$

where  $\lambda$  is a scalar non-negative smoothing parameter that controls the complexity of the spline fit. In a Bayesian modeling context, the spline coefficients,  $\mathbf{b}$ , can be viewed as random variables with a multivariate normal prior

$$p(\mathbf{b}) \propto \exp\left(-\frac{\mathbf{b}' \mathbf{S} \mathbf{b}}{\sigma_b^2}\right) \quad (7)$$

where  $\sigma_b^2$  is the parameter that controls the smoothness of the functional coefficient  $\beta(t)$ .

The prior distribution in (7) is easy to write, but its implementation may be subject to numeric instability, especially when multiple functional and scalar predictors are considered. To address this problem we use a linear reparametrization of the model parameters,  $\mathbf{b}$ , that corresponds to independent normal priors on the transformed parameters. This method was introduced by Ruppert et al. [31] in the context of semiparametric regression and was first implemented in a Bayesian context by Crainiceanu et al. [36] for penalized thin plate splines. The method was likely developed in parallel and extended by Wood and collaborators [22, 37, 38] in the context of multiple functions and penalty matrices that are not of full rank. Here we use a description of the method provided by Scheipl et al. [39] in the context of Bayesian analysis of additive regression models; the method was also used in the context of function-on-scalar Bayesian inference by Sun and Kowal [40].

Consider the spectral decomposition  $\mathbf{S} = \mathbf{U} \mathbf{V} \mathbf{U}'$ , where  $\mathbf{U}$  is a  $K \times K$  dimensional matrix such that  $\mathbf{U}' \mathbf{U} = \mathbf{U} \mathbf{U}' = \mathbf{I}_K$  and  $\mathbf{V}$  is a diagonal  $K \times K$  dimensional matrix. The columns of the matrix  $\mathbf{U}$  are the eigenvectors of the penalty matrix  $\mathbf{S}$ , while the diagonal elements of the matrix  $\mathbf{V}$  are its corresponding eigenvalues. If  $K_0$  is the rank of  $\mathbf{S}$ , we re-write  $\mathbf{S} = [\mathbf{U}_+ \mathbf{U}_0] \begin{bmatrix} \mathbf{V}_+ & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} [\mathbf{U}_+ \mathbf{U}_0]'$ , where  $\mathbf{U}_+$  is the  $K \times K_0$  matrix of eigenvectors associated with strictly positive eigenvalues,  $\mathbf{U}_0$  is the  $K \times (K - K_0)$  matrix of eigenvectors associated with the zero eigenvalues, and  $\mathbf{V}_+$  is the  $K_0 \times K_0$  dimensional diagonal matrix of non-zero eigenvalues. Consider the following reparametrization of the spline coefficients

$$\tilde{\mathbf{b}} = \tilde{\mathbf{V}}^{1/2} \mathbf{U}' \mathbf{b} \quad (8)$$

where  $\tilde{\mathbf{V}}^{1/2} = \begin{bmatrix} \mathbf{V}_+^{1/2} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{K-K_0} \end{bmatrix}$ ,  $\mathbf{V}_+^{1/2}$  is the diagonal matrix of square roots of the non-zero eigenvalues, and  $\mathbf{I}_{K-K_0}$  is the identity matrix

of rank  $K - K_0$ . With this notation we also have  $\mathbf{b} = \mathbf{U} \tilde{\mathbf{V}}^{-1/2} \tilde{\mathbf{b}}$ , which provides a way of recovering the original spline parameters,  $\mathbf{b}$ , from the transformed parameters,  $\tilde{\mathbf{b}}$ . This is important because working with  $\tilde{\mathbf{b}}$  is easier, but reconstructing the functional parameter  $\beta(t)$  requires  $\mathbf{b}$ .

We further partition  $\tilde{\mathbf{b}}$  into  $\tilde{\mathbf{b}} = (\tilde{\mathbf{b}}_r, \tilde{\mathbf{b}}_f)'$  where  $\tilde{\mathbf{b}}_r$  are the first  $K_0$  coefficients. From the definition,  $\tilde{\mathbf{b}}_r = [\mathbf{V}_+^{1/2} \mathbf{0}] \mathbf{U}' \mathbf{b}$  and

$$\tilde{\mathbf{b}}_r' \mathbf{I}_{K_0} \tilde{\mathbf{b}}_r = \mathbf{b}' \mathbf{U} ([\mathbf{V}_+^{1/2} \mathbf{0}])' [\mathbf{V}_+^{1/2} \mathbf{0}] \mathbf{U}' \mathbf{b} = \mathbf{b}' \mathbf{S} \mathbf{b} \quad (9)$$

Thus, using the penalty matrix  $\mathbf{S}$  for  $\mathbf{b}$  is equivalent to using the identity penalty matrix for  $\tilde{\mathbf{b}}_r$  with smoothing parameter  $\sigma_b^2$  and no penalty on  $\tilde{\mathbf{b}}_f$ .

To make the linear predictor  $\boldsymbol{\eta}$  consistent with the previous definition, we also need to transform the design matrix  $\mathbf{X}$  to  $\tilde{\mathbf{X}}$  such that  $\tilde{\mathbf{X}} \tilde{\mathbf{b}} = \mathbf{X}' \mathbf{b}$ . If  $\tilde{\mathbf{V}}^{-1/2}$  is the inverse of  $\tilde{\mathbf{V}}^{1/2}$  and

$$\tilde{\mathbf{X}} = \tilde{\mathbf{V}}^{-1/2} \mathbf{U}' \mathbf{X} \quad (10)$$

then  $\tilde{\mathbf{X}} \tilde{\mathbf{b}} = \mathbf{X} \mathbf{U} \tilde{\mathbf{V}}^{-1/2} \tilde{\mathbf{V}}^{1/2} \mathbf{U}' \mathbf{b} = \mathbf{X}' \mathbf{b}$ . We further partition  $\tilde{\mathbf{X}}$  into  $\tilde{\mathbf{X}} = [\tilde{\mathbf{X}}_r' \tilde{\mathbf{X}}_f']$ , where  $\tilde{\mathbf{X}}_r'$  are the first  $K_0$  columns of  $\tilde{\mathbf{X}}'$ . With this notation the full Bayesian SoFR model where the functional parameter  $\beta(t)$  is modeled nonparametrically using penalized splines is

$$\begin{cases} \mathbf{Y} \sim \text{Exponential Family}(\boldsymbol{\eta}, a); \\ \boldsymbol{\eta} = \eta_0 \mathbf{J}_n + \tilde{\mathbf{X}}_r' \tilde{\mathbf{b}}_r + \tilde{\mathbf{X}}_f' \tilde{\mathbf{b}}_f + \mathbf{Z}' \boldsymbol{\gamma}; \\ \tilde{\mathbf{b}}_r \sim N(\mathbf{0}, \sigma_b^2 \mathbf{I}); \\ \eta_0 \sim p(\eta_0); \tilde{\mathbf{b}}_f \sim p(\tilde{\mathbf{b}}_f); \boldsymbol{\gamma} \sim p(\boldsymbol{\gamma}); \\ \sigma_b^2 \sim p(\sigma_b^2); a \sim p(a) \end{cases} \quad (11)$$

where  $\boldsymbol{\eta}$  is the linear predictor and  $a$  is the dispersion parameter for the exponential family distribution. Here  $p(\cdot)$  is a general notation for uninformative priors on fixed effects. For example,  $p(\tilde{\mathbf{b}}_2)$  are  $K - K_0$  independent zero-mean normal priors with large variance (small precision) and  $p(\sigma_b^2)$  is an inverse Gamma prior  $IG(0.001, 0.001)$  [27, 36], though other non-informative variance priors could also be used. An alternative direct penalization approach was recently introduced by Sartini et al. [41], who proposed a fully Bayesian functional principal components analysis that projects eigenfunctions onto an orthonormal spline basis and efficiently samples the low-dimensional coefficient matrix via polar decomposition, with order constraints imposed during the sampling process to ensure identifiability. However, here we use the reparameterization described above.

The main contribution here is to transform the conceptual functional regression model (1) into the relatively simple Bayesian model (11) that can be easily implemented in Stan. This also provides the infrastructure for adding additional smoothing and mixed effects components that would be difficult to incorporate without this initial stepping stone.

## 2.3 | Bayesian SoFR Implementation in Stan

We now provide a detailed, step-by-step tutorial on how to implement the Bayesian SoFR model (11) using the R interface to Stan. We will introduce the data preparation, Stan program, and how to organize the results for statistical inference. Our associated R package `refundBayes` makes this process invisible to users, but we describe the underlying infrastructure for reasons related to reproducibility, transparency, and future software development. We provide the detailed code in the [Supporting Information](#). The `refundBayes` package relies on the `rstan` package for Bayesian computation. Please ensure that `rstan` is installed and loaded before using functions from `refundBayes`.

### 2.3.1 | Data Preparation

Assume that the functional data  $\{W_i(t_m), i = 1, \dots, n, m = 1, \dots, M\}$  is stored in R as an  $n \times M$  matrix `data$wmat`, where the  $i$ -th row contains  $M$  observations from subject  $i$ . To construct the matrix  $\mathbf{X}$  defined in (3), we store the time points in an  $n \times M$  matrix `data$tmat`, where the  $(i, m)$ -th entry is equal to  $t_{im} = t_m$ , as defined in Section 2.1. Additionally, let `data$lmat` be an  $n \times M$  matrix whose  $(i, m)$ -th entry equals  $L_m = t_{m+1} - t_m$ . The outcome is stored as a vector `data$Y`. As shown in Crainiceanu et al. [3], the frequentist SoFR can be fitted using the `mgcv` : : `gam` function as follows:

```
// Fit model using frequentist approaches: mgcv
fit = mgcv::gam(Y ~ s(tmat, by=lmat*wmat, bs="cr", k=10),
  data = data)
```

The argument `s(tmat, by=lmat*wmat, bs="cr", k=10)` constructs the matrix  $\mathbf{X}$  using cubic regression splines (specified in the argument `bs="cr"`) with a 10-dimensional basis (specified in the argument `k=10`). We will leverage this argument to construct the input for Stan. Specifically, calculating the matrices  $\tilde{\mathbf{X}}_r$  and  $\tilde{\mathbf{X}}_f$  in (11) can be done directly using the `smoothCon` function in the `mgcv` package:

```
// mgcv construction of design matrix for functional predictors
smcon = mgcv::smoothCon(s(tmat, by=lmat*wmat, bs="cr", k=10),
  data = data, absorb.cons = TRUE,
  diagonal.penalty = TRUE)
```

The `smoothCon` function first creates the spline basis according to the user-supplied spline information, including the spline type and the maximum number of degrees of freedom. It then integrates the basis with the functional covariate  $\mathbf{W}$  as described in (2). By setting the argument `diagonal.penalty = TRUE`, the program will perform basis transformation introduced in (10) and generate the design matrix with a diagonal penalty term. Here we set `absorb.cons = TRUE` so that the identifiability constraints are absorbed when constructing the spline basis. The matrices  $\tilde{\mathbf{X}}_r$  and  $\tilde{\mathbf{X}}_f$  can then be obtained using the `mgcv` : : `smooth2random` function as follows:

```
// Extract the transformed design matrices
randeff = mgcv::smooth2random(smcon[[1]], name=names(data), type
= 2)
X_mat_r = t(randeff$rand$Xr)
X_mat_f = t(randeff$rand$Xf)
```

In this section we took advantage of the built-in structure of the `mgcv` package. Users may prefer to build these components themselves, which can be done following the steps described in Section 2.2.

### 2.3.2 | Stan Code

The transformed design matrices  $\tilde{\mathbf{X}}_r$  and  $\tilde{\mathbf{X}}_f$  are stored in the data list as `data$X_mat_r` and `data$X_mat_f`, respectively, in addition to the outcome  $\mathbf{Y}$  stored in `data$Y` and the design matrix for scalar predictors  $\mathbf{Z}$  stored in `data$Z`. A typical Stan code consists of three main blocks: the `data` block, which defines the format of the input data; the `parameters` block, which specifies the parameters to be sampled; and the `model` block, which formulates the log-likelihood of the Bayesian model. The following `data` block defines the data components for the SoFR model:

```
// Stan code data block
data {
  int<lower=1> N_num; // Total number of subjects
  int Y[N_num]; // Outcome variable
  int<lower=1> K_num; // Number of scalar predictors
  // Design matrix for the scalar predictors
  matrix[K_num, N_num] Z_mat;
  int Kf; // Row number of the fixed effects design matrix
  int Kr; // Row number of the random effects design matrix
  matrix[Kf, N_num] X_mat_f; // Fixed effects design matrix
  matrix[Kr, N_num] X_mat_r; // Random effects design matrix
}
```

The parameters `N_num`, `K_num`, `Kf`, `Kr` are easily obtained from the model input.

The `parameters` block specifies the parameters to be sampled in Stan:

```
// Stan code parameter block
parameters {
  real sigma; // Smoothing parameter
  real eta_0; // Linear predictor intercept
  vector[Kf] betaf; // Fixed effects spline coefficients
  vector[Kr] betar; // Random effects spline coefficients
  vector[K_num] gamma; // Coefficients for scalar predictors
}
```

And finally, the `model` block specifies the joint log-likelihood, including the data likelihood and the priors for the parameters. Here we show the implementation for binary outcomes using the `bernoulli_logit_lpmf` function in Stan. However, the model can as easily be applied to other exponential family distributions by calling the corresponding Stan functions. The code below follows the Bayesian model described in equation (11), where `target` is the token for the logarithm of the likelihood in Stan:

```
// Stan code model block
model {
  // Linear predictor
  vector[N_num] eta = rep_vector(0.0, N_num);
  eta += eta_0 + X_mat_f' * betaf + X_mat_r' * betar + Z_mat' *
gamma;
  // Log-likelihood for the binary outcome
  for (n in 1:N_num) {
    target += bernoulli_logit_lpmf(Y[n] | eta[n]);
  }
  // Set priors for the parameters
  target += normal_lpdf(betar | 0, sigma);
  target += inv_gamma_lpdf(sigma^2 | 0.001, 0.001);
}
```

Here we do not specify explicit priors for the parameters `gamma`, `beta.f`, and `eta.0`. In this case, Stan implicitly assigns a uniform prior to these parameters over their respective domains. Another option is to specify noninformative priors for these parameters in the `model` block. No difference was noticed between the two approaches.

### 2.3.3 | Reconstructing the Estimated Functional Coefficients

Recall that  $\mathbf{b} = \mathbf{U}\tilde{\mathbf{V}}^{-1/2}\tilde{\mathbf{b}}$  and  $\beta(t) = \sum_{k=1}^K b_k \psi_k(t)$ . For every iteration of the algorithm, we obtain a sample from the joint posterior distribution of the  $K \times 1$  dimensional vector of parameters  $\tilde{\mathbf{b}} = \{\tilde{b}_1, \dots, \tilde{b}_K\}$ . Hence, at all times  $t$ , simultaneously, we obtain a sample of the functional coefficient as follows

$$\tilde{\mathbf{b}} \rightarrow \mathbf{b} = \mathbf{U}\tilde{\mathbf{V}}^{-1/2}\tilde{\mathbf{b}} \rightarrow \beta(t) = \sum_{k=1}^K b_k \psi_k(t)$$

Again, these quantities can be built from scratch, but we used the `mgcv` infrastructure to compute them and construct the spline basis. However, the transformation matrix  $\tilde{\mathbf{V}}$  used for orthogonalization and scaling is not directly provided in the output of `mgcv::smoothCon`. To map the posterior samples of  $\tilde{\mathbf{b}}$  back to the original spline basis and reconstruct  $\beta(t)$ ,  $\tilde{\mathbf{V}}$  is computed from scratch. We start by extracting the spline basis  $\boldsymbol{\psi}(t)$  and the corresponding penalty matrix  $\mathbf{S}$  using the `mgcv::smooth.construct` function.

```
// Construct the spline smoother components
splinecons = mgcv::smooth.construct(object, data, NULL)
```

Here `object` is the `mgcv::s` term that specifies the spline type, parameter, and functional predictor. For example, for the cubic regression splines with 10 degrees of freedom the `object` term would be `s(tmat, by=lmat*wmat, bs="cr", k=10)`. The spline basis  $\boldsymbol{\psi}(t)$  can be extracted as:

```
// Extract spline basis
Psi_mat = splinecons$X
```

The corresponding penalty matrix  $\mathbf{S}$  can be extracted as:

```
// Extract penalty matrix
S_mat = splinecons$S
```

The rank,  $K_0$ , of the penalty matrix  $\mathbf{S}$  can be extracted as:

```
// Extract the rank of S
rank = splinecons$rank
```

As  $\mathbf{S} = \mathbf{U}\mathbf{V}\mathbf{U}'$ , the  $\mathbf{U}$  and  $\mathbf{V}$  matrices are obtained as:

```
// Spectral decomposition
eigendecom = eigen(S_mat, symmetric = TRUE)
// Eigenfunctions
U_mat = eigendecom$vectors
// Eigenvalues
V_vec = eigendecom$value[1:rank]
```

Now we calculate the diagonal elements of the  $\tilde{\mathbf{V}}^{1/2}$  matrix used in the `smoothCon` function:

```
V = rep(1, ncol(X_mat))
// First K0 entries
V[1:rank] = sqrt(V_vec)
// Calculate the remaining K-K0 entries
col.norm = colSums((X_mat %*% U_mat)^2)
col.norm = col.norm / V2
av.norm = mean(col.norm[1:rank])
for (i in (rank + 1):ncol(X_mat)) {
  V[i] = sqrt(col.norm[i] / av.norm)
}
```

Here `X_mat` is the transposed of the original design matrix  $\mathbf{X}$ . The `smoothCon` function uses a slightly different definition for the  $K \times K$  matrix  $\tilde{\mathbf{V}}$  than what is presented in (8). Specifically, both the `smoothCon` function and equation (8) define the first  $K_0$  entries of the diagonal vector of the matrix  $\tilde{\mathbf{V}}$  as the positive eigenvalues of  $\mathbf{S}$ . However, unlike (8), where the remaining  $K - K_0$  diagonal entries are constant and equal to 1, the `smoothCon` function defines them based on the norm of the design matrix  $\mathbf{X}$  (as illustrated in the code). This exact construction of  $\tilde{\mathbf{V}}$  is necessary if `mgcv` is used to construct the model components, but other options could be used, as well, as long as one keeps track of the reparameterization operators.

As  $\mathbf{b} = \mathbf{U}\tilde{\mathbf{V}}^{-1/2}\tilde{\mathbf{b}}$ , the posterior samples of the spline coefficients,  $b^q$ ,  $q = 1, \dots, Q$ , can be obtained as:

```
// Transform the posterior sample of the spline coefficient
beta.sample.untilde = (U_mat %*% diag(1 / V)) %*% beta.sample
```

Here `beta.sample` contains the posterior sample of the transformed spline coefficients  $\tilde{\mathbf{b}}$ . Having all necessary ingredients, the posterior sample of the functional effect  $\beta(t)$  can be calculated at every iteration,  $q = 1, \dots, Q$ , as

$$\beta^q(t) = \sum_{k=1}^K b_k^q \psi_k(t), \quad q = 1, \dots, Q \quad (12)$$

The posterior samples of the functional effect  $\beta^q(t)$  are stored in a  $Q \times M$  dimensional matrix, where each row corresponds to a posterior iteration and each column corresponds to a location,  $t$ , on the functional domain. The column mean,  $\hat{\beta}(t)$ , and the covariance of this matrix,  $\hat{\mathbf{C}}_\beta$ , can then be used to provide pointwise and correlation and multiplicity adjusted (CMA) credible intervals; see [3] for an in-depth description of CMA. Pointwise confidence intervals can also be obtained using pointwise quantiles (separately at every  $t$ ), which can be especially useful when the posterior distribution of  $\beta(t)$  is not Gaussian. In the supplementary R Markdown file, we provide the complete code for our Bayesian Stan model.

## 2.4 | Simulations

Using simulations, we compare the inferential accuracy of the Bayesian SoFR implementation with that of the state-of-the-art frequentist approach.

### 2.4.1 | Data Generating Mechanism

We consider the total of  $n = 100, 200, 300$ , and 500 subjects and one functional covariate observed at  $T = 50$  equally-spaced

points in  $[0, 1]$ . The functional covariates  $\{W_i(t_j), t_j \in [0, 1]\}_{i=1}^T$  are generated using the first 4 principal components estimated from our NHANES case study [42]. The functional coefficient is  $\beta(t) = (0.084 - (t - 0.5)^2) \times \tau$ ,  $t \in [0, 1]$ , with  $\tau = 1, 2, 3$ , and 5 controlling the strength of the signal. The outcome variables  $Y_i, i = 1, \dots, n$  are generated from either a Gaussian distribution with mean  $\eta_i = \frac{1}{T} \sum_{j=1}^T W_i(t_j) \beta(t_j)$  and standard deviation of 1.5, or a Bernoulli distribution with success probability  $\exp(\eta_i) / \{1 + \exp(\eta_i)\}$ .

### 2.4.2 | Competing Methods

We compare the performance of our Bayesian Stan program with the results obtained using the `gamfunction` from the R package `mgcv`. For both Bayesian and frequentist methods, we use a cubic regression spline (`bs="cr"`) basis with a maximum of 10 degrees of freedom (`k=10`). As described in Crainiceanu et al. [3], one can also use the `refund::pfr` [11] function in R to conduct the analyses.

### 2.4.3 | Metrics

For each simulation scenario, we calculate the relative integrated squared error (RISE) of the estimated functional effect, defined as:  $\int \{\beta(t) - \hat{\beta}(t)\}^2 dt / \int \beta^2(t) dt$ , where  $\beta(t)$  and  $\hat{\beta}(t)$  are the true and the estimated functional coefficient, respectively. We compare the average empirical coverage rate for the pointwise 95% confidence intervals. In addition to evaluating the coefficients, we also assess the predictive performance on a new set of 500 study participants by comparing the predicted outcomes to the corresponding true values.

### 2.4.4 | Results

Simulation results are presented in Table 1 (for Gaussian outcomes) and Table 2 (for binary outcomes), based on 500 simulations for each scenario. We report the median relative integrated

squared error (RISE) and the mean coverage rate. As sample size and signal level vary, the median RISE ranges from near zero (indicating high estimation accuracy) to values above 5 (indicating that the estimation is far from the truth). Although the estimation performance varies across scenarios, the Bayesian approach consistently demonstrates similar performance with the frequentist approach for both Gaussian and binary outcomes. Both methods exhibit similar RISE, with the Bayesian approach achieving a slightly higher coverage rate of the pointwise confidence intervals. The Supporting Information provides the mean computation time (in minutes) for the Bayesian Stan program.

## 3 | Bayesian Functional Cox Regression

In this section, we describe the process of fitting a Bayesian Functional Cox Regression (FCR) model with a time-to-event outcome in Stan. Surprisingly, there are few published frequentist approaches for analyzing such data; see, for example, Gellar et al. [43]; Kong et al. [44]; Qu et al. [45], who proposed different versions of the “linear functional Cox model.” Recently, Cui et al. [46] introduced the Additive Functional Cox Model (AFCM), which extended the methods introduced by Gellar et al. [43] to account for non-linear functional effects, as introduced by McLean et al. [47] for generalized functional regression models. Extending these methods to Bayesian analysis is not straightforward, as a full likelihood needs to be specified, including modeling of the baseline hazard function. In this tutorial, we model the baseline hazard function using splines [48, 49], and we have adopted a new combination of priors to ensure a robust performance of the Stan implementation.

### 3.1 | The Functional Cox Regression Model

For subject  $i = 1, \dots, n$ , denote by  $T_i$  the event time and by  $C_i$  the censoring time, where  $T_i$  is observed only when  $T_i \leq C_i$ .

**TABLE 1** | Simulation results for scalar-on-function regression (SoFR) comparing the frequentist and Bayesian approach for Gaussian outcomes for different sample sizes,  $n$ , and signal levels,  $\tau$ .

		$n = 100$		$n = 200$		$n = 300$		$n = 500$	
		Bayes	Freq	Bayes	Freq	Bayes	Freq	Bayes	Freq
$\tau = 1$	RISE	4.694	6.025	2.431	2.973	1.837	2.123	1.367	1.55
	Coverage	(96.4)	(94.5)	(96.7)	(94.3)	(97.1)	(93.8)	(96.3)	(92.8)
	Prediction	3.653	5.785	1.8	2.926	1.365	2.112	1.017	1.459
$\tau = 2$	RISE	1.525	1.725	0.917	0.925	0.623	0.554	0.394	0.352
	Coverage	(95.9)	(92.9)	(97.1)	(95.1)	(97.2)	(95.2)	(97.9)	(95.9)
	Prediction	1.121	1.827	0.691	0.863	0.457	0.555	0.278	0.339
$\tau = 3$	RISE	0.892	0.842	0.413	0.353	0.298	0.287	0.154	0.151
	Coverage	(96.5)	(93.5)	(97.4)	(96)	(98.9)	(97.6)	(99.5)	(98.4)
	Prediction	0.683	0.814	0.294	0.349	0.196	0.253	0.103	0.142
$\tau = 5$	RISE	0.334	0.314	0.148	0.148	0.111	0.103	0.067	0.063
	Coverage	(98.3)	(96.6)	(99.4)	(97.5)	(99.5)	(98)	(99.8)	(98.9)
	Prediction	0.239	0.293	0.1	0.132	0.069	0.094	0.042	0.056

Note: The median RISE, mean coverage rate of the 95% credible/confidence intervals, and prediction accuracy are reported.

**TABLE 2** | Simulation results for Bernoulli outcomes using the same structure and measures as Table 1.

		$n = 100$		$n = 200$		$n = 300$		$n = 500$	
		Bayes	Freq	Bayes	Freq	Bayes	Freq	Bayes	Freq
$\tau = 1$	RISE	9.792	11.087	4.529	5.577	3.117	3.611	2.081	2.311
	Coverage	(96.4)	(95.7)	(96)	(94.3)	(96.8)	(94.7)	(96.3)	(93.5)
	Prediction	6.618	10.737	3.101	5.497	2.141	3.486	1.499	2.357
$\tau = 2$	RISE	2.722	2.932	1.525	1.687	1.031	1.1	0.71	0.634
	Coverage	(95.6)	(94.3)	(95.8)	(93.1)	(96.3)	(94)	(97.7)	(96)
	Prediction	1.862	2.925	1.155	1.607	0.765	1.055	0.514	0.633
$\tau = 3$	RISE	1.425	1.525	0.829	0.782	0.604	0.521	0.324	0.287
	Coverage	(96.3)	(93.7)	(96.7)	(94.3)	(97.5)	(95.6)	(98.6)	(97.2)
	Prediction	1.084	1.47	0.626	0.721	0.417	0.473	0.214	0.279
$\tau = 5$	RISE	0.748	0.667	0.34	0.313	0.213	0.198	0.126	0.117
	Coverage	(97.2)	(95.9)	(98.5)	(97)	(99)	(97.9)	(99.3)	(97.8)
	Prediction	0.539	0.61	0.234	0.264	0.138	0.19	0.087	0.108

The observed data is  $[Y_i, \delta_i, \mathbf{Z}_i, \{W_i(t_{im}), t_{im} \in [0, 1]\}]$ , where  $Y_i = \min(T_i, C_i)$ , and  $\delta_i$  is the indicator of right censoring with  $\delta_i = 0$  if  $Y_i = T_i$  (an observed event), and  $\delta_i = 1$  if  $Y_i < T_i$  (a censored event),  $\mathbf{Z}_i$  is the  $p \times 1$  dimensional vector of scalar predictors, and  $\{W_i(t_{im}), t_{im} \in [0, 1]\}$  for  $m = 1, \dots, M_i$  is the functional predictor.

If  $T$  is a random variable, its hazard function is defined as the instantaneous rate of occurrence for the event at time  $t$ :

$$h_T(t) = \lim_{\Delta t \rightarrow 0} \frac{\Pr(t \leq T \leq t + \Delta t | T > t)}{\Delta t} \quad (13)$$

The cumulative hazard function is defined as  $H_T(t) = \int_{u=0}^t h(u) du$ , the cumulative distribution function (cdf) is defined as  $F_T(t) = P(T \leq t)$ , and the survival function is defined as  $S_T(t) = 1 - F_T(t)$ . It can easily be shown that  $S_T(t) = \exp\{-H(t)\}$ . To model the individual hazard function we consider a natural extension of the Cox proportional hazards model [50], which assumes that  $h_i(t) = h_0(t) \exp(\eta_i)$ , where  $h_0(t)$  is the baseline hazard function, and  $\eta_i$  is the linear predictor for subject  $i$  defined in equation (4). The log-likelihood for this model has the following form

$$l(\mathbf{Y}, \delta; h_0, \boldsymbol{\eta}) = \sum_{i=1}^n [(1 - \delta_i) [\log\{h_0(y_i)\} + \eta_i - H_0(y_i) \exp(\eta_i)] + \delta_i \{-H_0(y_i) \exp(\eta_i)\}] \quad (14)$$

where  $H_0(t) = \int_{u=0}^t h_0(u) du$  is the cumulative baseline hazard function.

### 3.2 | Model of the Hazard Function Using M-splines

Similar to Brilleman et al. [48], we model the baseline hazard function  $h_0(t)$  using an M-spline basis [51], such that  $h_0(t) = \sum_{l=1}^L c_l M_l(t; \mathbf{k}, \tau)$ , where  $M_l(t; \mathbf{k}, \tau)$  denotes the  $l$ -th M-spline basis with knots  $\mathbf{k}$  and degree of freedom  $\tau$ , and  $\mathbf{c} = (c_1, \dots, c_L)'$  are the spline coefficients. We require that  $c_l \geq 0$  and  $\sum_{l=1}^L c_l > 0$ , which ensures that  $h_0(t) \geq 0$  and  $h_0(t) \neq 0$  (events do occur).

M-splines are a family of non-negative, piecewise polynomial basis functions integrate to one over their support. To model the cumulative baseline hazard function, we use I-splines, which are the integrated forms of M-splines and therefore yield monotone functions.

If  $I_l(t; \mathbf{k}, \tau) = \int_0^t M_l(u; \mathbf{k}, \tau) du$  denotes the corresponding I-spline basis function [51], the cumulative baseline hazard function  $H_0(t) = \sum_{l=1}^L c_l I_l(t; \mathbf{k}, \tau)$  is non-decreasing by construction.

The I-spline coefficients,  $\mathbf{c}$ , and the intercept,  $\eta_0$ , in the linear predictor  $\eta_i$  are not simultaneously identifiable. Indeed, for any  $a > 0$  the parameters  $(\mathbf{c}, \eta_0)$  and  $(\tilde{\mathbf{c}}, \tilde{\eta}_0) = \{a\mathbf{c}, \eta_0 - \log(a)\}$  correspond to the same hazard  $h_0(t) \exp(\eta_i)$ . To make the model identifiable, we impose the constraint  $\sum_{l=1}^L c_l = 1$  by assuming that the coefficients  $\mathbf{c}$  have a non-informative Dirichlet prior  $D(\mathbf{c}; \boldsymbol{\alpha})$ , where  $\boldsymbol{\alpha} = (1, \dots, 1)$ . The full Bayesian functional Cox regression model is

$$\begin{cases} \mathbf{Y} \sim l(\mathbf{Y}, \delta; h_0, \boldsymbol{\eta}); \\ \boldsymbol{\eta} = \eta_0 \mathbf{J}_n + \tilde{\mathbf{X}}_r' \tilde{\mathbf{b}}_1 + \tilde{\mathbf{X}}_f' \tilde{\mathbf{b}}_2 + \mathbf{Z}' \boldsymbol{\gamma}; \\ \tilde{\mathbf{b}}_1 \sim N(\mathbf{0}, \sigma_b^2 \mathbf{I}); \\ \eta_0 \sim p(\eta_0); \tilde{\mathbf{b}}_2 \sim p(\tilde{\mathbf{b}}_2); \boldsymbol{\gamma} \sim p(\boldsymbol{\gamma}); \sigma_b^2 \sim p(\sigma_b^2); \\ h_0(t) = \sum_{l=1}^L c_l M_l(t; \mathbf{k}, \tau); \\ \mathbf{c} \sim D(\mathbf{c}; \boldsymbol{\alpha}) \end{cases} \quad (15)$$

where most components are similar to model (11), except the likelihood (first line of the model) and the specification of the baseline hazard using I-splines (last two lines of the model).

### 3.3 | Bayesian FCR Implementation in Stan

#### 3.3.1 | Data Preparation

Suppose that the data  $\mathbf{Y}$  are stored as a vector  $\mathbf{y}$  in  $\mathbb{R}$ , and the event indicator  $1 - \delta$  is stored in `data$event`. We first obtain the M-spline basis  $\mathbf{M} = \{M_1(t; \mathbf{k}, \tau), \dots, M_L(t; \mathbf{k}, \tau)\}$  using, for example, the `splines2::mSpline` [52, 53] function in R.

```
// Construct the M-spline basis
Mbasis = splines2::mSpline(y, Boundary.knots=c(min.bound,
max.bound), df=5, intercept=TRUE)
```

The lower bound of the knots is set to be slightly less than the minimal value of  $Y$ , and the upper bound is set to be slightly larger than the maximum value of  $Y$ . Here, we choose the degrees of freedom of the spline basis to be  $L = 5$  by setting  $df=5$ . The other parameters are automatically determined by the `mSpline` function. The corresponding I-spline basis can be obtained similarly by calling the `iSpline` function.

```
// Construct the I-spline basis
Ibasis = splines2::iSpline(y, Boundary.knots=c(min.bound,
max.bound), df=5)
```

### 3.3.2 | Stan Code

The Stan program for the linear functional Cox model is shown below. Specifying the log-likelihood requires both the baseline hazard rate and the linear predictor. Therefore, we need to add the censoring indicator variable, the M-spline basis, and the I-spline basis to the `data` block, in addition to those defined in Section 2.3:

```
// Stan code data block
data{
  .....
  array[N_num] int cens;
  int L_num;
  matrix[N_num, L_num] Mbasis;
  matrix[N_num, L_num] Ibasis;
}
```

Here `cens` is the censoring indicator  $\delta_i, i = 1, \dots, n$ , and `L_num` is the number of M-spline basis, `L`. `Mbasis` is an  $n \times L$  dimensional matrix, with the  $(i, l)$ -th entry equal to  $M_l(Y_i; \mathbf{k}, \tau)$ ; similarly `Ibasis` is the  $n \times L$  dimensional matrix corresponding to the integrated basis.

The `parameters` block, in addition to those defined in Section 2.3, includes the basis coefficients for the M-splines and I-splines, `c`:

```
// Stan code parameter block
parameters{
  .....
  simplex[L_num] c;
}
```

The spline coefficients `c` are defined as a simplex variable because its entries are non-negative and sum to one. The baseline hazard and the cumulated baseline hazard function share the same parameters and they can be calculated as `Mbasis*c` and `Ibasis*c`, respectively.

The `model` block defines several functions required for calculating the log-likelihood in (14).

```
// Stan code function block
// eta: linear predictor
// bhaz and cbhaz: baseline and cumulative baseline hazard
functions {
  real cox_log_lhaz(real y, real eta, real bhaz, real cbhaz)
  {return log(bhaz) + eta;}
  real cox_log_lccdf(real y, real eta, real bhaz, real cbhaz)
  {return - cbhaz * exp(eta);}
  real cox_log_lpdf(real y, real eta, real bhaz, real cbhaz)
  {return cox_log_lhaz(y, eta, bhaz, cbhaz) +
    cox_log_lccdf(y | eta, bhaz, cbhaz);}
}
```

The code for the `model` block is shown below, where the log-likelihood is calculated using the functions defined in the `functions` block:

```
// Stan code model block
model {
  // Construct the baseline hazard rate and cumulated baseline
  hazard rate
  vector[N_num] bhaz = Mbasis * c;
  vector[N_num] cbhaz = Ibasis * c;
  // Linear predictor
  vector[N_num] eta = rep_vector(0.0, N_num);
  eta += eta_0 + X_mat_f' * betaf + X_mat_r' * betar + Z_mat' *
  gamma;
  // Log-likelihood for the time-to-event outcome
  for (n in 1:N_num) {
    if (cens[n] == 0) {
      target += cox_log_lpdf(Y[n] | eta[n], bhaz[n],
cbhaz[n]);
    }
    else if (cens[n] == 1) {
      target += cox_log_lccdf(Y[n] | eta[n], bhaz[n],
cbhaz[n]);
    }
  }
  // Set priors for the parameters
  vector[L_num] alpha = rep_vector(1, L_num);
  target += dirichlet_lpdf(c | alpha);
  target += normal_lpdf(betar | 0, sigma);
  target += inv_gamma_lpdf(sigma2 | 0.001, 0.001);
}
```

As in Section 2, we omit an explicit prior expression for the parameters `gamma`, `betaf`, and `eta_0`, which are then set automatically by Stan.

## 3.4 | Simulations

Similar to SoFR, we compare the performance of our Bayesian Functional Cox Regression model implemented in Stan with the frequentist implementation in the `mgcv` : `gam` function. The `mgcv` package handles Cox proportional hazards models through its `cox.ph()` family, which fits the model using partial likelihood and a Breslow-type nonparametric estimator of the baseline hazard function.

### 3.4.1 | Data Generating Mechanism

We use a simulation procedure similar to that introduced in Section 2.4. Following Cui et al. [46], we generate the survival outcome based on our case study NHANES data. More specifically, we generate the baseline hazard function according to the estimated baseline hazard function from NHANES. The hazard function for each subject can then be calculated using the generated baseline hazard function and the simulated linear predictors. Once the hazard function is calculated, event times are simulated according to the one-to-one relationship between the hazard function and the distribution function. The censoring time for each subject is randomly sampled among the simulated event times; see the [Supporting Information](#) for a complete implementation of the simulations.

### 3.4.2 | Results

The simulation results are presented in Table 3. Consistent with the results for SoFR, our proposed Bayesian algorithm has a good

**TABLE 3** | Simulation results for functional Cox regression comparing the frequentist and Bayesian approach for time-to-event outcomes for different sample sizes,  $n$ , and signal levels,  $\tau$ .

		$n = 100$		$n = 200$		$n = 300$		$n = 500$	
		Bayes	Freq	Bayes	Freq	Bayes	Freq	Bayes	Freq
$\tau = 1$	RISE	4.281	5.053	2.288	2.52	1.763	1.838	1.261	1.247
	Coverage	(96.3)	(96.2)	(96.9)	(95.7)	(96.1)	(95.3)	(96.2)	(95.6)
	Prediction	3.095	3.29	1.637	1.874	1.284	1.366	0.9	0.981
$\tau = 2$	RISE	1.626	1.728	0.814	0.801	0.601	0.519	0.362	0.297
	Coverage	(94.8)	(94.5)	(96.7)	(96.9)	(97.3)	(97.8)	(98.4)	(98.6)
	Prediction	1.169	1.303	0.613	0.557	0.413	0.351	0.241	0.213
$\tau = 3$	RISE	0.852	0.841	0.417	0.359	0.24	0.201	0.146	0.126
	Coverage	(95.9)	(95.9)	(97.9)	(98.3)	(98.7)	(99.1)	(99.2)	(99.4)
	Prediction	0.664	0.578	0.283	0.247	0.164	0.148	0.095	0.09
$\tau = 5$	RISE	0.3	0.276	0.146	0.128	0.099	0.092	0.061	0.058
	Coverage	(98.2)	(98.9)	(99)	(99.5)	(99.4)	(99.5)	(99.7)	(99.6)
	Prediction	0.207	0.196	0.095	0.089	0.063	0.059	0.041	0.039

Note: The median RISE, mean coverage rate of the 95% credible/confidence intervals, and prediction accuracy are reported.

performance in terms of the relative integrated squared error (RISE) and coverage rate compared with the frequentist method.

#### 4 | Bayesian Joint Functional Models With FPCA

In the models presented in Sections 2 and 3, the observed data are directly used as functional predictors in the regression framework. While this approach is widely adopted in the functional data analysis literature, the observed functional data are often measured with error and/or not measured at the same locations. A common strategy to address these challenges is to first apply functional principal component analysis (FPCA) to the observed data, obtain the predicted values on a grid, and use these predictions as functional predictors in the regression model. This two-step approach treats the FPCA predictions as fixed inputs to the regression, ignoring the uncertainty of the score estimators, which could affect estimation accuracy and coverage of the confidence intervals. To address these potential issues, we adopt a joint modeling approach that simultaneously estimates the PC scores and fits the regression model [54]. The approaches are similar for exponential family and time-to-event outcomes. Therefore, we only describe how to do this for time-to-event outcomes.

#### 4.1 | The Joint Functional Models With FPCA

##### 4.1.1 | FPCA Model and Notation

Assume that the observed functional covariate  $W_i(t)$  has the following structure  $W_i(t) = \mu(t) + D_i(t) + \epsilon_i(t)$ , where  $D_i(t)$  is a mean 0 stochastic process with covariance operator  $K_D(t, s) = \text{cov}\{D_i(t), D_i(s)\}$ , and  $\epsilon_i(t)$  is a white noise process. For simplicity, we assume that  $\mu(t) = 0$  and by Kosambi-Karhunen-Loève (KKL) theorem [55],  $D_i(t) = \sum_{j=1}^{\infty} \xi_{ij} \phi_j(t)$ , where  $\phi_j(t)$  are orthonormal eigenfunctions and  $\xi_{ij}$  are mutually uncorrelated random

variables with eigenvalues  $\lambda_j$ , respectively, where  $\lambda_j \geq 0$  is a decreasing sequence. Assuming that the first  $J$  eigenfunctions provide a good approximation for  $D_i(t)$ , it follows that  $W_i(t) \approx \sum_{j=1}^J \xi_{ij} \phi_j(t) + \epsilon_i(t)$ . A standard simplifying assumption is that  $\xi_{ij} \sim N(0, \lambda_j)$  and  $\epsilon_i(t) \sim N(0, \sigma_\epsilon^2)$ .

##### 4.1.2 | The Joint Functional Model

The joint model considers the regression of outcomes on the latent trajectory  $D_i(t)$  and not on the observed functions  $W_i(t)$ . The difference is that  $W_i(t)$  is observed with noise (sometimes referred to as “measurement error” for functional data), which could influence the point estimators and confidence intervals. To address this problem we consider models of the type

$$\begin{cases} Y_i \sim l(Y_i; \eta_i); \\ \eta_i = \eta_0 + \int_0^1 D_i(t) \beta(t) dt + \mathbf{Z}_i^T \boldsymbol{\gamma}; \\ W_i(t) = D_i(t) + \epsilon_i(t) \end{cases} \quad (16)$$

where  $l(Y_i; \eta_i)$  is the conditional distribution of the outcome  $Y_i$  given the linear predictor  $\eta_i$ , and could correspond to either the exponential or the time-to-event families of distributions. Other conditional distributions could also be considered, but are not addressed in this tutorial.

As  $\beta(t) = \sum_{k=1}^K b_k \psi_k(t)$  and  $D_i(t) = \sum_{j=1}^J \xi_{ij} \phi_j(t)$ , the linear predictor  $\eta_i$  can be expressed as

$$\begin{aligned} \eta_i &= \eta_0 + \sum_{j=1}^J \sum_{k=1}^K \xi_{ij} b_k \int_0^1 \phi_j(t) \psi_k(t) dt + \mathbf{Z}_i^T \boldsymbol{\gamma} \\ &= \eta_0 + \sum_{j=1}^J \sum_{k=1}^K \xi_{ij} b_k X_{jk} + \mathbf{Z}_i^T \boldsymbol{\gamma} \\ &= \eta_0 + \boldsymbol{\xi}_i^T \mathbf{X}^t \mathbf{b} + \mathbf{Z}_i^T \boldsymbol{\gamma} \end{aligned} \quad (17)$$

where  $X_{jk} = \int_0^1 \phi_j(t) \psi_k(t) dt$  is the inner product of the eigenfunction  $\phi_j(t)$  and the spline basis  $\psi_k(t)$ , and  $\mathbf{X}^t$  is the  $J \times K$  dimensional matrix with  $X_{jk}$  as the  $(j, k)$ -th entry. We incorporate penalization to the functional effect  $\beta(t)$  using the prior (7) on the spline coefficient  $\mathbf{b}$ . To simplify the penalization prior, we apply the same reparametrization technique as in Section 2.2. Keeping the same notation, the Bayesian joint SoFR accounting for measurement error around the observed signal becomes:

$$\left\{ \begin{array}{l} \mathbf{Y} \sim l(\mathbf{Y}, \delta; h_0, \boldsymbol{\eta}) ; \\ \boldsymbol{\eta} = \eta_0 \mathbf{J}_n + \xi \tilde{\mathbf{X}}_r \tilde{\mathbf{b}}_r + \xi \tilde{\mathbf{X}}_f \tilde{\mathbf{b}}_f + \mathbf{Z}^t \boldsymbol{\gamma} ; \\ h_0(t) = \sum_{l=1}^L c_l M_l(t; \mathbf{k}, \tau) ; \\ W_i(t) = \sum_{j=1}^J \xi_{ij} \phi_j(t) + \epsilon_i(t) ; \\ \xi_{ij} \sim N(\hat{\xi}_{ij}, \lambda_j), \lambda_j \sim p(\lambda_j), j = 1, \dots, J ; \\ \epsilon_i(t) \sim N(0, \sigma_\epsilon^2), \sigma_\epsilon^2 \sim p(\sigma_\epsilon^2); \mathbf{c} \sim p(\mathbf{c}) ; \\ \eta_0 \sim p(\eta_0); \tilde{\mathbf{b}}_r \sim p(\tilde{\mathbf{b}}_r); \tilde{\mathbf{b}}_f \sim p(\tilde{\mathbf{b}}_f) ; \\ \boldsymbol{\gamma} \sim p(\boldsymbol{\gamma}); \sigma_b^2 \sim p(\sigma_b^2) \end{array} \right. \quad (18)$$

where  $\hat{\xi}_{ij}$  are the frequentist estimates of the scores using FPCA,  $p(\lambda_j)$  and  $p(\sigma_\epsilon^2)$  are non-informative priors for the variance components, such as  $IG(0.001, 0.001)$ , though other priors could also be considered. The priors for  $\tilde{\mathbf{b}}_f$  and  $\tilde{\mathbf{b}}_r$  and  $\boldsymbol{\gamma}$  are the same as in Section 2, while the prior on  $\mathbf{c}$  is the Dirichlet prior with parameter  $\boldsymbol{\alpha} = (1, \dots, 1)$  described in Section 3.

## 4.2 | Bayesian Joint Functional Modeling in Stan

### 4.2.1 | Data Preparation

We first apply the frequentist FPCA to the observed functional covariate data `$wmat` using the `refund::fpcfa.face` function in R:

```
// Fit the functional PCA model
fpcfa.fit = refund::fpcfa.face(Y=data$wmat)
```

The estimated eigenfunctions can be extracted from `fpcfa.fit$efunctions`. Similar to Section 2, we use the `mgcv::smooth.construct` function in R to construct the spline basis:

```
// Construct raw spline basis
splinecons = mgcv::smooth.construct(object, data, NULL)
```

where `object` is the `mgcv` term that specifies the spline type, parameter, and functional predictor. For example, for the cubic regression splines with 10 degrees of freedom, as described in Section 2.3, the `object` term would be `s(tmat, by=1mat*wmat, bs="cr", k=10)`. The spline basis  $\boldsymbol{\psi}(t)$  can be extracted as:

```
// Extract the raw spline basis
Psi_mat = splinecons$X
```

The corresponding penalty matrix  $\mathbf{S}$  can be extracted as:

```
// Extract the penalty matrix
S_mat = splinecons$S
```

We then calculate the  $\mathbf{X}$  matrix in (17) by integrating the eigenfunction with the spline basis:

```
// Calculate the design matrix
X_mat_t = matrix(nrow=J_num, ncol=K_num)
for(j in 1:J_num){
  for(k in 1:K_num) {
    X_mat_t[j,k] = sum(fpcfa.fit$efunctions[,j] * Psi_mat[,k])
  }
}
/ M_num
}
```

where `X_mat_t` refers to the transpose of  $\mathbf{X}$  matrix, `J_num`, `K_num` are the corresponding values of  $J, K$ , and `M_num` is the number of functional observations  $M$ . To transform the matrix  $\mathbf{X}$  to  $\tilde{\mathbf{X}}$ , we first perform the spectral decomposition of the extracted penalty matrix `S_mat`:

```
// Spectral decomposition of the design matrix
eigendecomp = eigen(S_mat, symmetric = TRUE)
```

The transformation is done using the following code:

```
// Obtain the design matrix
rank = splinecons$rank
E = rep(1, ncol(X_mat))
E[1:rank] = sqrt(eigendecomp$value[1:rank]) // Square root of eigenvalues
X_mat_t = X_mat_t %*% eigendecomp$vectors
X_mat_t = t(t(X_mat_t) / E)
```

where `rank` is the rank of the penalty matrix, which is equal to the number of random effects basis. The transformed random effects design matrix  $\tilde{\mathbf{X}}_r$  can be extracted as:

```
// Extract the random effects design matrix
X_mat_r = t(X_mat_t[,1:rank])
```

and the transformed fixed effect design matrix  $\tilde{\mathbf{X}}_f$  is

```
// Extract the fix effect design matrix
X_mat_f = t(X_mat_t[, (rank + 1):ncol(X_mat_t)])
```

### 4.2.2 | Stan Code

We now describe the Stan code for our Bayesian joint SoFR model with time-to-event outcomes. We skip here the data and parameter blocks, though they are provided in the [Supporting Information](#). Instead, we focus on the `model` block and identify the main changes from Sections 2 and 3. The `model` block is:

```
// Stan code model block
model {
  // Baseline and cumulative baseline hazard functions
  vector[N_num] bhaz = Mbasis * c;
  vector[N_num] cbhaz = Ibasis * c;
  // Linear predictor
  vector[N_num] eta = rep_vector(0.0, N_num);
```

```

eta += eta_0 + xi * X_mat_f' * betaf + xi * X_mat_r' * betar
+ Z_mat' * gamma;
// Likelihood for the time-to-event outcome
for (n in 1:N_num) {
  if (cens[n] == 0) {
    target += cox_log_lpdf(Y[n] | eta[n], bhaz[n],
cbhaz[n]);
  }
  else if (cens[n] == 1) {
    target += cox_log_lcdf(Y[n] | eta[n], bhaz[n],
cbhaz[n]);
  }
}
// Likelihood for the functional observation in FPCA
target += - N_num * M_num * log(sigma_e) - sum((xi * Phi_mat
- M_mat)^2) / (2 * sigma_e^2);
// Set prior for the FPCA scores xi
for(nj in 1:J_num){
  target += - N_num * log(lambda[nj]) -
sum((xi[,nj] - xi_hat[,nj])^2) / (2 * lambda[nj]^2);
}
// Other priors
vector[L_num] alpha = rep_vector(1, L_num);
target += dirichlet_lpdf(c | alpha);
target += normal_lpdf(betar | 0, sigma);
target += inv_gamma_lpdf(sigma2|0.001,0.001);
for(nj in 1:J_num){
  target += inv_gamma_lpdf(lambda[nj]|0.001,0.001);
}
target += inv_gamma_lpdf(sigma_e2|0.001,0.001);
}

```

The priors for other parameters  $\eta_{a,0}$ ,  $\beta_{af}$ , and  $\gamma$  are omitted in the Stan model, effectively imposing uniform priors as discussed in Section 2.

## 4.3 | Simulations

### 4.3.1 | Data Generating Mechanism

Simulation experiments follow the same scenarios described in Sections 2.4 and 3.4. The only difference is that the observed functional data,  $\mathbf{W}$ , are now simulated based on the estimated eigenfunctions from the data application but with added independent errors  $\epsilon_i(t_j) \sim N(0, \sigma_\epsilon^2)$ . We consider two cases, one with moderate noise,  $\sigma_\epsilon = 5$ , and one with large noise,  $\sigma_\epsilon = 10$ . Details on simulations are provided in the [Supporting Information](#).

### 4.3.2 | Competing Methods

We compare the performance of our Bayesian Stan program with the results obtained using the `gam` function from the R package `mgcv`. For both Bayesian and frequentist methods, we use the cubic regression spline (`bs="cr"`) with 10 degrees of freedom (`k=10`) to fit the model. As described in Crainiceanu et al. [3], one key distinction between the two approaches is that the frequentist approach conditions on  $\xi_{ij}$ , whereas the joint Bayesian approach incorporates the uncertainty associated with estimating these scores.

### 4.3.3 | Results

Table 4 provides the simulation results, which indicate that the joint Bayesian algorithm: (1) has similar performance in terms of relative integrated squared error (RISE); and (2) achieves

superior coverage, especially when the noise is large, likely due to accounting for the variability of the score predictions.

## 5 | Bayesian Function-on-Scalar Regression

In this section, we provide the step-by-step tutorial for implementing the Bayesian Function-on-Scalar Regression (FoSR) model using Stan. The procedure for fitting a generalized multilevel Bayesian FoSR was introduced in Goldsmith et al. [56]; the primary purpose of this section is to provide a comprehensive software solution using Stan.

### 5.1 | The Bayesian FoSR Model

Let  $Y_i(t)$  be the functional response for subject  $i = 1, \dots, n$  and at time points  $t = t_1, \dots, t_M$ . FoSR assumes the following model for the functional response:

$$Y_i(t) = \sum_{p=1}^P X_{ip} \beta_p(t) + e_i(t) \quad (19)$$

where  $X_{ip}$  with  $p = 1, \dots, P$  are the scalar predictors, and  $\beta_p(t)$  are the corresponding domain-varying (or functional) coefficients. The first covariate is often the intercept,  $X_{i1} = 1$ . Two key differences between FoSR and traditional regression are that: (1) the outcome,  $Y_i(t)$ , is multivariate and often high-dimensional; and (2) the residuals  $e_i(t)$  are correlated across  $t$  (points in the domain). To account for these data features, we assume that the residuals  $e_i(t)$  have a zero-mean Gaussian Process (GP) distribution, but we also indicate how the software can be modified for non-Gaussian outcomes (e.g., binary, Poisson). Specifically, we assume that  $e_i(t) = \sum_{j=1}^J \xi_{ij} \phi_j(t) + \epsilon_i(t)$ , where  $\phi_1(t), \dots, \phi_J(t)$  are the eigenfunctions and  $\epsilon_i(t)$  are assumed to be independent  $N(0, \sigma_\epsilon^2)$ . With this notation, the model becomes:

$$\begin{aligned}
Y_i(t) &= \sum_{p=1}^P X_{ip} \beta_p(t) + \sum_{j=1}^J \xi_{ij} \phi_j(t) + \epsilon_i(t) \\
&= \sum_{p=1}^P X_{ip} \sum_{k=1}^K b_{pk} \psi_k(t) + \sum_{j=1}^J \xi_{ij} \phi_j(t) + \epsilon_i(t) \\
&= \sum_{k=1}^K \left( \sum_{p=1}^P X_{ip} b_{pk} \right) \psi_k(t) + \sum_{j=1}^J \xi_{ij} \phi_j(t) + \epsilon_i(t) \quad (20)
\end{aligned}$$

Denote by  $B_{ik} = \sum_{p=1}^P X_{ip} b_{pk}$ , by  $\mathbf{B}_i = (B_{i1}, \dots, B_{iK})'$  the  $K \times 1$  dimensional vector with  $B_{ik}$  as the  $k$ -th entry, by  $\boldsymbol{\psi}(t) = \{\psi_1(t), \dots, \psi_K(t)\}$  the  $1 \times K$  dimensional vector of spline coefficients evaluated at time  $t$ , by  $\boldsymbol{\Psi}$  the  $M \times K$  dimensional matrix with the  $m$  row equal to  $\boldsymbol{\psi}(t_m)$ , by  $\boldsymbol{\phi}(t) = \{\phi_1(t), \dots, \phi_J(t)\}$  the  $1 \times J$  dimensional vector of spline coefficients evaluated at time  $t$ , and by  $\boldsymbol{\Phi}$  the  $M \times J$  dimensional matrix with the  $m$  row equal to  $\boldsymbol{\phi}(t_m)$ . With this notation, we have that  $\sum_{k=1}^K (\sum_{p=1}^P X_{ip} b_{pk}) \psi_k(t) = \boldsymbol{\psi}(t) \mathbf{B}_i$ ,  $\sum_{j=1}^J \xi_{ij} \phi_j(t) = \boldsymbol{\phi}(t) \boldsymbol{\xi}_i$ , and  $Y_i(t) = \boldsymbol{\psi}(t) \mathbf{B}_i + \boldsymbol{\phi}(t) \boldsymbol{\xi}_i + \epsilon_i(t)$ . If  $\mathbf{Y}_i = \{Y_i(t_1), \dots, Y_i(t_M)\}'$  is the  $M \times 1$  dimensional vector of observed data for study participant  $i$  and  $\boldsymbol{\epsilon}_i = \{\epsilon_i(t_1), \dots, \epsilon_i(t_M)\}'$  is the  $M \times 1$  vector of random errors, we have  $\mathbf{Y}_i = \boldsymbol{\Psi} \mathbf{B}_i + \boldsymbol{\Phi} \boldsymbol{\xi}_i + \boldsymbol{\epsilon}_i$ . Let  $\mathbf{b}_p = (b_{p1}, \dots, b_{pK})'$  denote the  $K \times 1$  dimensional vector of spline coefficients for  $\beta_p(t)$ , and  $\mathbf{S}$  denote the smoothing penalty

**TABLE 4** | Simulation results comparing the two-step approach that conditions on FPCA scores with the joint Bayesian (Stan) methods for functional predictors with time-to-event outcomes for different number of observations,  $n$ , and signal levels,  $\tau$ .

			$n = 100$		$n = 200$		$n = 300$		$n = 500$	
			Bayes	Freq	Bayes	Freq	Bayes	Freq	Bayes	Freq
Median variance	$\tau = 1$	RISE	4.739	4.665	2.742	2.725	1.8	1.922	1.297	1.356
		Coverage	(96.6)	(94.5)	(96.3)	(93)	(96.9)	(93.4)	(95.9)	(91.7)
		Prediction	3.103	3.478	1.89	2.057	1.295	1.44	0.911	1.024
	$\tau = 2$	RISE	1.704	1.752	0.954	0.986	0.698	0.607	0.42	0.346
		Coverage	(94.5)	(90.8)	(95.7)	(92)	(96.3)	(93.8)	(97)	(95.2)
		Prediction	1.156	1.265	0.685	0.67	0.457	0.407	0.245	0.209
	$\tau = 3$	RISE	0.913	0.958	0.508	0.383	0.315	0.255	0.221	0.179
		Coverage	(94.5)	(91.6)	(96.6)	(95.1)	(97.4)	(96.5)	(97.9)	(97)
		Prediction	0.703	0.661	0.331	0.27	0.198	0.173	0.115	0.105
	$\tau = 5$	RISE	0.434	0.332	0.232	0.189	0.165	0.143	0.11	0.096
		Coverage	(95.5)	(94.3)	(97.8)	(97.3)	(97.9)	(97.1)	(98.1)	(98)
		Prediction	0.265	0.23	0.116	0.11	0.085	0.078	0.055	0.053
Large variance	$\tau = 1$	RISE	5.788	5.807	2.782	2.833	1.912	1.984	1.293	1.29
		Coverage	(97.6)	(94)	(97.2)	(92.3)	(97.6)	(91.5)	(97.7)	(90.7)
		Prediction	3.939	4.12	1.842	2.092	1.43	1.545	0.94	1.081
	$\tau = 2$	RISE	1.771	1.853	0.999	1.042	0.815	0.777	0.503	0.37
		Coverage	(97.2)	(91.7)	(96.7)	(89.6)	(96.5)	(89.2)	(97.7)	(93.2)
		Prediction	1.261	1.348	0.786	0.859	0.664	0.602	0.361	0.277
	$\tau = 3$	RISE	1.061	1.111	0.619	0.479	0.418	0.298	0.23	0.175
		Coverage	(96.2)	(88.3)	(96.1)	(91.3)	(97.3)	(94)	(98.5)	(96.8)
		Prediction	0.807	0.922	0.446	0.338	0.288	0.218	0.143	0.115
	$\tau = 5$	RISE	0.578	0.449	0.259	0.192	0.186	0.141	0.126	0.094
		Coverage	(95.8)	(90.6)	(97.8)	(95.6)	(98.4)	(96.5)	(99.2)	(98.2)
		Prediction	0.464	0.327	0.171	0.142	0.129	0.107	0.08	0.065

Note: The median RISE, mean coverage rate of the 95% credible/confidence intervals, and prediction accuracy are reported.

structure for  $\mathbf{b}_p$  as described in Section 2. The Bayesian FoSR model can then be represented as

$$\begin{cases}
 Y_i = \Psi \mathbf{B}_i + \Phi \xi_i + \epsilon_i ; \\
 B_{ik} = \sum_{p=1}^P X_{ip} b_{pk}, k = 1, \dots, K ; \\
 p(\mathbf{b}_p) \propto \exp(-\mathbf{b}'_p \mathbf{S} \mathbf{b}_p / 2\sigma_p^2), p = 1, \dots, P ; \\
 \xi_{ij} \sim N(0, \lambda_j), \lambda_j \sim p(\lambda_j), j = 1, \dots, J ; \\
 \epsilon_i(t_m) \sim N(0, \sigma_\epsilon^2), i = 1, \dots, n, t = t_1, \dots, t_M ; \\
 \sigma_\epsilon^2 \sim p(\sigma_\epsilon^2), \sigma_p^2 \sim p(\sigma_p^2), p = 1, \dots, P
 \end{cases} \quad (21)$$

We impose the normal smoothing priors  $p(\mathbf{b}_p)$  by specifying them in the model. One could apply the SVD techniques described in Sections 2 and 3, but we use the original spline basis without transformation because: (1) this approach was adopted in Goldsmith et al. [56], and (2) using the untransformed basis simplifies interpretation and avoids the need to back-transform the estimated coefficients. The priors  $p(\lambda_j)$ ,  $p(\sigma_\epsilon^2)$ , and  $p(\sigma_p^2)$  are all non-informative priors on variance components; here we use inverse Gamma priors  $IG(0.001, 0.001)$ , but other priors could also be considered. Notice that every function  $\beta_p(t)$  has its own smoothing parameter,  $\sigma_p^2$ , which allows for fixed effects

parameters to have different levels of smoothing (more or fewer degrees of freedom.) In model (21), we assume that the functional responses are observed on a common set of grid points across all subjects. This assumption ensures that the  $\Psi$  matrix is identical across subjects, which simplifies matrix operations in Stan. However, the FoSR model is, in principle, applicable when functional responses are observed on subject-specific grids. Developing efficient Bayesian algorithms for such settings remains an important direction for future work.

## 5.2 | Bayesian FoSR Implementation in Stan

### 5.2.1 | Data Preparation

As illustrated in Section 4, the eigenfunctions  $\Phi$  can be obtained by applying FPCA to the functional response  $Y_i(t)$  using the frequentist `refund::fPCA.face` function. Assume that `data$yindex` is a  $M \times n$  index matrix with the  $i$ -th column being the sequence  $(t_1, \dots, t_M)$ . The spline basis  $\Psi$  and the corresponding penalty matrix  $\mathbf{S}$  can be obtained using the `smooth.construct` function:

```
// Construct the spline basis
object = s(yindex, bs = "cr", k = 10)
splinecons = mgcv::smooth.construct(object, data, NULL)
```

where `object` is the `mgcv` term that specifies the spline information as described in Section 4. Notice that the `object` here is different from other sections because we now have functional responses. The spline basis  $\Psi$  can be extracted as:

```
// Extract the spline basis
Psi_mat = splinecons$X
```

The penalty matrix  $\mathbf{S}$  can be extracted as:

```
// Extract the penalty matrix
S_mat = splinecons$S
```

## 5.2.2 | Stan Code

We now describe the Stan code for the Bayesian FoSR. The data block is presented below:

```
// Stan code data block
data {
  int<lower=1> N_num; // Total number of participants
  int<lower=1> M_num; // Total number of observed functional
  time point
  matrix[N_num, M_num] Y_mat; // Functional response
  int<lower=1> P_num; // Number of scalar predictors
  matrix[N_num, P_num] X_mat; // Design matrix for the scalar
  predictor
  int<lower=1> J_num; // Number of FPCA eigenfunctions
  matrix[J_num, M_num] Phi_mat; // Matrix of FPCA
  eigenfunctions
  int<lower=1> K_num; // Number of spline basis
  matrix[K_num, M_num] Psi_mat; // Matrix of spline basis
  matrix[K_num, K_num] S_mat; // Penalty matrix
}
```

The parameter block is:

```
// Stan code parameter block
parameters {
  matrix[K_num, P_num] beta; // Spline coefficients
  matrix[N_num, J_num] xi; // FPCA Scores
  real<lower=0> sigma_eps; // Standard deviation of independent
  error
  vector<lower=0>[P_num] sigma; // Smoothing parameter
  vector<lower=0>[J_num] lambda; // FPCA eigenvalues
}
```

The model block is:

```
// Stan code model block
model {
  matrix[N_num, M_num] mu;
  // Fitted mean matrix
  mu = X_mat * beta' * Psi_mat + xi * Phi_mat;
  // Log-likelihood for functional response
  target += - N_num * M_num * log(sigma_eps) / 2 - sum((mu -
  Y_mat)^2) / (2 * sigma_eps^2);
  // Prior for the penalized spline coefficients
  for(np in 1:P_num){
    target += (- beta[np] * S_mat * beta[np]) / (2 *
    sigma[np]^2);
    target += inv_gamma_lpdf(sigma[np]^2|0.001,0.001);
  }
  // Prior for the FPCA scores
  for(nj in 1:J_num){
    target += - N_num * log(lambda[nj]) / 2 - sum((xi[nj]
    ^2) / (2 * lambda[nj]^2));
    target += inv_gamma_lpdf(lambda[nj]^2|0.001,0.001);
  }
  // Other priors
  target += inv_gamma_lpdf(sigma_eps^2|0.001,0.001);
}
```

Since  $\xi$  and the remaining part of  $\beta$  have non-informative priors, we omit their explicit expressions in the Stan model.

## 5.3 | Simulations

### 5.3.1 | Data Generating Mechanism

Here we adapt the simulation setting from the previous sections and add one scalar predictor  $X_i \sim N(20, 10)$  for  $i = 1, \dots, n$  with  $n = 100, 300, 500, 700$ . The functional responses are generated from the model  $Y_i(t) = X_i\beta(t) + W_i(t) + \epsilon_i(t)$ , where  $\beta(t) = \{0.084 - (t - 0.5)^2\} \times \tau$  and  $\tau = 0.5, 1, 2, 4$ . The random effects  $W_i(t)$  are generated as in Section 2.4 using the principal components, and  $\epsilon_i(t)$  are generated independently from  $N(0, 5)$ . As usual, the total number of time points  $M$  is set to 50 and we compare the relative integrated squared error (RISE) of the Bayesian approach with that of the frequentist results using the `mgcv` package.

### 5.3.2 | Results

The simulation results are presented in Table 5. The Bayesian FoSR algorithm has a similar performance with `mgcv`.

## 6 | NHANES Data Applications

We apply the Bayesian functional regression programs to quantify the association between scalar and functional predictors and mortality in the National Health and Nutrition Examination Survey (NHANES). The functional predictor is the minute-level average daily physical activity measured using accelerometers. The outcome is a binary indicator of 5-year all-cause mortality. Crainiceanu et al. [3] conducted a SoFR using the NHANES dataset to examine the association between physical activity and all-cause mortality.

In this section, our Bayesian analysis results are compared to those reported in Crainiceanu et al. [3].

### 6.1 | NHANES Physical Activity Study

NHANES is a nationwide study conducted by the United States Centers for Disease Control and Prevention (CDC) to assess the health and nutritional status of adults and children in the United States [57]. It is conducted in two-year waves with approximately 10,000 participants per wave. For the purpose of this paper, we focus on data collected from the NHANES 2011-2012 and NHANES 2013-2014 waves, which included up to seven days of continuous physical activity data collected using a tri-axial wrist-worn accelerometer [58]. Accelerometry data was summarized by NHANES at the minute level using the “Monitor Independent Movement Summary” (MIMS) units [59]. The distribution of MIMS at each time point exhibits substantial skewness and a log-transformation was applied at the minute level, which resulted in more symmetric marginal distributions. Based on previous studies [3, 14, 46], for every individual, physical activity data were averaged over available days of that individual at each minute, resulting in a 1,440-dimensional vector of average log-MIMS values for every study participant. Processing

TABLE 5 | Simulation results for FoSR.

		$n = 100$		$n = 300$		$n = 500$		$n = 700$	
		Bayes	Freq	Bayes	Freq	Bayes	Freq	Bayes	Freq
$\tau = 0.5$	RISE	0.0085	0.0067	0.0032	0.0034	0.0022	0.0025	0.0018	0.0021
	Coverage	(99.85)	(95.11)	(99.61)	(94.44)	(99.2)	(93.88)	(98.93)	(93.49)
$\tau = 1$	RISE	0.0025	0.0029	0.0012	0.0017	0.0009	0.0015	0.0008	0.0013
	Coverage	(99.42)	(94.2)	(98.21)	(92.3)	(96.94)	(88.98)	(95.71)	(86.59)
$\tau = 2$	RISE	0.001	0.0015	0.0007	0.0012	0.0006	0.0011	0.0006	0.0011
	Coverage	(98.12)	(90.96)	(94.85)	(82.79)	(92.29)	(76.97)	(90.26)	(73.6)
$\tau = 4$	RISE	0.0006	0.0011	0.0006	0.001	0.0005	0.001	0.0005	0.0010
	Coverage	(91.49)	(79.42)	(84.56)	(67.37)	(78.75)	(60.88)	(73.75)	(55.5)

Note: Results are structured as in all other simulation tables.

was conducted using a similar pipeline to that described in the `rnhanesdata` package [60]. Mortality was determined by linking the NHANES data to death certificate records from the National Death Index, maintained by the National Center for Health Statistics (NCHS), through the end of 2019.

In this case study, the outcome is a binary indicator of 5-year all-cause mortality. Although the NHANES study provides additional covariates, we limited our adjustment to sociodemographic factors as in Crainiceanu et al. [3]. The predictors are the minute-level physical activity expressed in average log-MIMS, age, gender, race, body mass index (BMI), poverty-to-income ratio (PIR), coronary heart disease (CHD), and education level. The data after preprocessing can be downloaded from [http://www.ciprianstats.org/sites/default/files/nhanes/nhanes\\_fda\\_with\\_r.rds](http://www.ciprianstats.org/sites/default/files/nhanes/nhanes_fda_with_r.rds)

The following code loads the dataset:

```
// Load the NHANES dataset
nhanes_use = readRDS("nhanes_fda_with_r.rds")
```

The NHANES 2011-2012 and 2013-2014 waves include a total of 12,610 participants with accelerometry data. Participants with missing outcome (3897 participants) and scalar covariates (an additional 1107 participants) were excluded. One additional participant was excluded because their coronary heart disease (CHD) status was recorded as “Refused”. After these exclusions, the final dataset used for this analysis included 7605 participants. The following code provides the data exclusion process from the `nhanes_use` data set:

```
# Names of the scalar predictors
pred.names = c("age", "gender", "race", "BMI", "PIR", "CHD",
              "education")
# Only include participants with fully observed scalar
predictors and outcome
for(cova in pred.names){
  nhanes_use = nhanes_use[!is.na(nhanes_use[cova]),]
}
nhanes_use = nhanes_use[!is.na(nhanes_use$event),]
nhanes_use = nhanes_use[-which(nhanes_use$CHD=="Refused"),]
# Construct the grid matrix and time point matrix for functional
predictor
nhanes_use$lmat = I(matrix(1/1440, ncol=1440,
nrow=nrow(nhanes_use)))
nhanes_use$tmat = I(matrix(1:1440, ncol=1440,
nrow=nrow(nhanes_use), byrow=TRUE))
```

## 6.2 | Frequentist Approach Based on Mixed Effects Representation

We fit the scalar-on-function regression model described in (1) for a binary outcome  $Y_i$  (an indicator of death during the follow-up period), functional predictor  $W_i(t)$  (average physical activity at time  $t$  from midnight), and covariates  $Z_i$  (described in Section 6.1). We first fit SoFR using the frequentist approach [3, 11] and implemented using the `mgcv::gam` function; as described in Crainiceanu et al. [3], this can be implemented in `refund::pfr` [21]. Since the functional covariate is periodic (midnight is both 0 and 24), we model the functional effect,  $\beta(t)$ , as periodic using a cyclic cubic penalized spline. The frequentist model based on mixed effects representation of SoFR can be fit using the following code:

```
// Frequentist model based on mixed effects representation
library(mgcv)
fit_freq_b = gam(event ~ age + gender + race + BMI + PIR + CHD +
education + s(tmat, by=lmat*MIMS, bs="cc", k=10),
data=nhanes_use, family=binomial())
```

The estimated functional and scalar coefficients as well as their confidence intervals can be obtained through the `mgcv::plot.gam` and `mgcv::summary` functions:

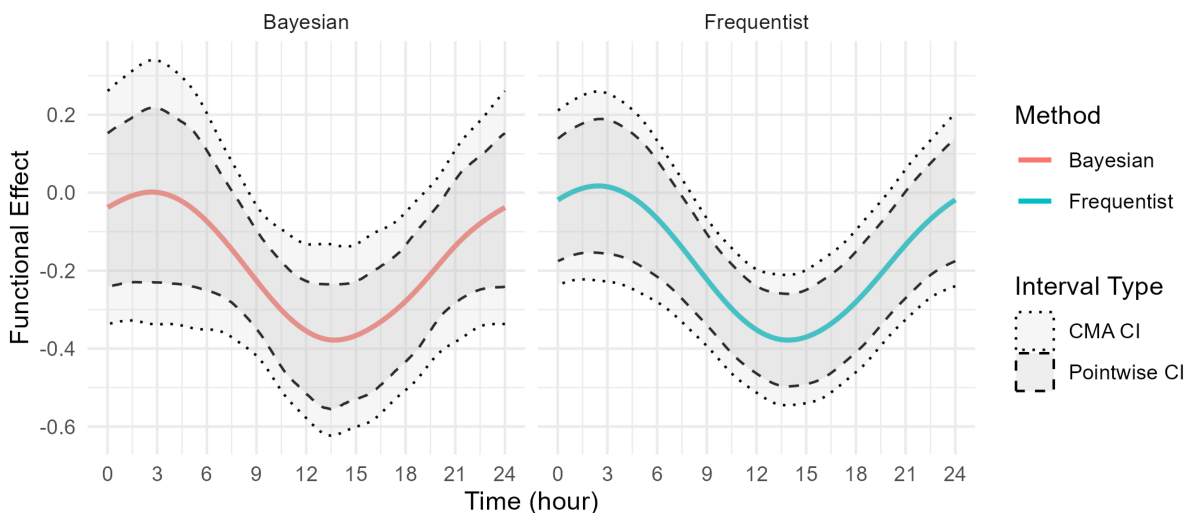
```
// Estimated functional coefficient
plot.gam(fit_freq_b)
// Estimated scalar coefficients
summary(fit_freq_b)
```

Results for the functional coefficient are presented in Figure 1 (right panel, labeled “Frequentist”). In Table 6, we present the estimated functional regression coefficients.

## 6.3 | Bayesian Approach Using Stan

### 6.3.1 | Bayesian Model Fit

We now analyze the same model using the Bayesian `Stan` program introduced in this paper on the same dataset. Our R package `refundBayes` was developed to have a similar syntax to that in `mgcv::gam`, though it uses Bayesian posterior inference and a few additional arguments that are specific to the



**FIGURE 1** | Estimated functional effect for the scalar-on-function regression for Bayesian (left) and Frequentist (right) methods. Darker gray shaded area bordered by dashed lines: pointwise 95% confidence/credible interval. Lighter gray shaded area bordered by dotted lines: CMA 95% confidence/credible interval.

**TABLE 6** | Estimated coefficients for scalar predictors in the NHANES example with corresponding 95% confidence/credible intervals for the frequentist and Bayesian implementation.

Scalar covariate	Bayesian estimate	Frequentist estimate
Age	0.063 (0.054, 0.073)	0.063 (0.054, 0.072)
Gender (female)	-0.068 (-0.280, 0.148)	-0.066 (-0.278, 0.146)
BMI	-0.023 (-0.039, -0.007)	-0.023 (-0.039, -0.007)
PIR	-0.165 (-0.243, -0.088)	-0.163 (-0.240, -0.087)
Race		
Other Hispanic	0.115 (-0.472, 0.720)	0.105 (-0.480, 0.690)
Non-Hispanic White	0.649 (0.201, 1.127)	0.626 (0.164, 1.089)
Non-Hispanic Black	0.454 (-0.023, 0.955)	0.432 (-0.050, 0.914)
Non-Hispanic Asian	-0.259 (-0.921, 0.400)	-0.256 (-0.918, 0.405)
Other Race	0.734 (-0.061, 1.500)	0.738 (-0.037, 1.514)
CHD		
Yes	0.568 (0.255, 0.872)	0.569 (0.263, 0.874)
Don't know	1.331 (0.335, 2.301)	1.324 (0.376, 2.271)
Education		
High school equivalent	-0.146 (-0.429, 0.138)	-0.143 (-0.424, 0.137)
More than high school	-0.370 (-0.642, -0.094)	-0.368 (-0.639, -0.097)
Don't know	2.199 (-0.428, 5.592)	1.811 (-0.692, 4.315)

Stan implementation. The eventual goal is for the user to simply run the program without implementing it from scratch.

The `bfrs` function uses the same data format as the `mgcv::gam` function, including the spline construction. The following code shows how `refundBayes` package fits the Bayesian SoFR using Stan:

```
// Bayesian approach using refundBayes package
library(refundBayes)
fit_bfrs = bfrs(event ~ age + gender + race + BMI + PIR + CHD +
  education + s(tmat, by=lmats*MIMS, bs="cc", k=10),
  data=nhanes_use, family=binomial(), joint_FPCA=F,
  n.iter=15000, n.warmup=5000, n.knots=3)
```

For the `bfrs` function, the `data` and the `family` arguments specify the dataset and the outcome type, which have the same

meaning as in the `gam` function. The `joint_FPCA` argument specifies whether to simultaneously perform FPCA within a joint model, as described in Section 5. The `n.iter`, `n.warmup`, and `n.knots` arguments are the total number of iterations, the burn-in value, and the number of chains (and knots for parallel computing) for posterior samples, respectively. The Stan program in this example used 15,000 posterior samples with 5000 burn-in iterations using 3 different chains.

We use  $k = 10$  number of basis in this example for illustrative purposes. In practical Bayesian analysis, the choice of the number of basis functions involves a trade-off between model flexibility and computational cost. Too few basis functions may fail to capture the true functional signal, while too many can result in unstable estimates and higher computational burden, even when penalization on the spline coefficients is applied. We recommend selecting the basis dimension by considering the complexity of the underlying functions, the resolution of the observed data, and the sample size. As a general guideline, 30-40 basis functions are often sufficient for moderately smooth functional data observed on dense grids [61].

### 6.3.2 | Results

The function `plot.bfrs(fit_bfrs)` displays the fitted functional coefficients with both the pointwise and the correlation and multiplicity adjusted (CMA) credible intervals. The CMA credible interval is constructed in a similar way as the CMA confidence interval described in section 2.4.3 of Crainiceanu et al. [3]. Specifically, denote by  $\beta^q(t)$  the  $q$ -th posterior sample of the functional effect,  $\beta(t)$ . Then, we can calculate the posterior samples from the max statistic  $d^q = \max\{|\beta^q(t) - \hat{\beta}(t)|/\hat{S}(t) : t = t_1, \dots, t_M\}$  where  $\hat{\beta}(t_i)$  and  $\hat{S}(t_i)$  is the mean and standard deviation of the posterior sample  $\{\beta^q(t_i)\}_{q=1}^Q$ , respectively. Then, the upper and lower bound of the CMA credible interval can be calculated as  $\hat{\beta}(t) \pm q_{d,1-\alpha} \times \hat{S}(t)$ , where  $q_{d,1-\alpha}$  is the 100(1 -  $\alpha$ ) percentile of the distribution of  $\{d^q, q = 1, \dots, Q\}$ .

The Bayesian estimated functional coefficient and 95% credible intervals are displayed in the left panel of Figure 1. The pointwise credible intervals are displayed as a darker shade of gray bordered by dashed lines, while the CMA credible intervals are displayed as a lighter shade of gray bordered by dotted lines. Compared with the estimators for the frequentist approach displayed in the right panel of the same figure, we conclude that the shape and length of confidence intervals are largely comparable, though the Bayesian credible intervals tend to be about 21% wider. This could be due to accounting for the uncertainty of the smoothing parameter, though more investigation would be necessary to identify the sources of this variation. For both Bayesian and frequentist approaches, the CMA credible interval is roughly 32% wider than their pointwise counterparts. Results for the scalar coefficients are presented in the Table 6 (labeled “Bayes”), which indicates an excellent agreement with the frequentist results.

### 6.3.3 | Bayesian Model Diagnosis

Assessing the convergence of Markov Chain Monte Carlo (MCMC) algorithms is a critical step in Bayesian modeling. One commonly used diagnostic tool for evaluating convergence is the traceplot, which displays the sampled values of a parameter across iterations of the MCMC algorithm. An ideal traceplot will show convergence with the parameter values oscillating around the mode of the posterior distribution [62]. Visually inspection using the `traceplot` function from `Rrstan` package is displayed in Figure 1 of the [Supporting Information](#) and indicates good convergence.

## 7 | Discussion

We introduced a Bayesian `Stan` program for scalar-on-function, functional Cox, and function-on-scalar regression models. This work builds upon the core ideas for functional regression described in Crainiceanu et al. [3]: model functional effects parametrically or nonparametrically using splines, penalize the spline coefficients, and conduct inference in the resulting mixed effects models. These ideas have a long history with previous Bayesian implementations [27, 36] in `WinBUGS` [28]. However, the emergence of the powerful `Stan` [24] Bayesian package provides substantial computational improvements and seamless integration with `R` [23]. Simulation results indicate that our specific implementation has reasonable inferential properties and compares well with existing approaches.

More work will need to be done to address computational stability and sensitivity to prior choices, but the approach is fully reproducible; see the [Supporting Information](#). When repeated measures of functional data are available (e.g., over seven days in our example), investigating the variability across those measurements is of interest. One natural approach is to apply multilevel functional principal component analysis (MFPCA). Our Bayesian joint modeling framework in Section 4 can be easily extended to MFPCA settings, although a full investigation of its statistical properties is left for future work.

### Acknowledgments

The authors thank the reviewers for their constructive comments, which greatly improved the article. This work was partially supported by the

NIH grants R01 NS060910 (CC) and R01 AG075883 (CC). This work represents the opinions of the researchers and not necessarily that of the granting organizations.

### Conflicts of Interest

The authors declare no conflicts of interest.

### Data Availability Statement

The data that support the findings of this study are openly available in Functional Data Analysis with R: Datasets at [https://functionaldataanalysis.org/dataset\\_nhanes.html](https://functionaldataanalysis.org/dataset_nhanes.html).

### References

1. J. Ramsay and B. Silverman, *Functional Data Analysis* (Springer, 2005).
2. P. Kokoszka and M. Reimherr, *Introduction to Functional Data Analysis* (CRC Press, 2017).
3. C. M. Crainiceanu, J. Goldsmith, A. Leroux, and E. Cui, *Functional Data Analysis With R* (CRC Press, 2024).
4. R. Sergazinov, A. Leroux, E. Cui, et al., “A Case Study of Glucose Levels During Sleep Using Multilevel Fast Function on Scalar Regression Inference,” *Biometrics* 79, no. 4 (2023): 3873–3882.
5. V. Baladandayuthapani, B. K. Mallick, M. Young Hong, J. R. Lupton, N. D. Turner, and R. J. Carroll, “Bayesian Hierarchical Spatially Correlated Functional Data Analysis With Application to Colon Carcinogenesis,” *Biometrics* 64, no. 1 (2008): 64–73.
6. C. Z. Di, C. M. Crainiceanu, B. S. Caffo, and N. M. Punjabi, “Multilevel Functional Principal Component Analysis,” *Annals of Applied Statistics* 3, no. 1 (2009): 458–488.
7. S. Salvatore, J. G. Bramness, M. J. Reid, K. V. Thomas, C. Harman, and J. Røislien, “Wastewater-Based Epidemiology of Stimulant Drugs: Functional Data Analysis Compared to Traditional Statistical Methods,” *PLoS One* 10, no. 9 (2015): e0138669.
8. G. Loewinger, E. Cui, D. M. Lovinger, and F. Pereira, “A Statistical Framework for Analysis of Trial-Level Temporal Dynamics in Fiber Photometry Experiments,” *Elife* 13 (2025): RP95802.
9. X. Leng and H. G. Müller, “Classification Using Functional Data Analysis for Temporal Gene Expression Data,” *Bioinformatics* 22, no. 1 (2006): 68–76.
10. J. Peng, D. Paul, and H. G. Müller, “Time-Warped Growth Processes, With Applications to the Modeling of Boom–Bust Cycles in House Prices,” *Annals of Applied Statistics* 8, no. 3 (2014): 1561–1582.
11. J. Goldsmith, J. Bobb, C. M. Crainiceanu, B. Caffo, and D. Reich, “Penalized Functional Regression,” *Journal of Computational and Graphical Statistics* 20, no. 4 (2011): 830–851.
12. J. Goldsmith, C. M. Crainiceanu, B. Caffo, and D. Reich, “Longitudinal Penalized Functional Regression for Cognitive Outcomes on Neuronal Tract Measurements,” *Journal of the Royal Statistical Society: Series C: Applied Statistics* 61, no. 3 (2012): 453–469.
13. P. Reiss and R. Ogden, “Functional Generalized Linear Models With Images as Predictors,” *Biometrics* 66, no. 1 (2010): 61–69.
14. E. Cui, A. Leroux, E. Smirnova, and C. M. Crainiceanu, “Fast Univariate Inference for Longitudinal Functional Models,” *Journal of Computational and Graphical Statistics* 31, no. 1 (2022): 219–230.
15. R. Li, L. Xiao, E. Smirnova, E. Cui, A. Leroux, and C. M. Crainiceanu, “Fixed-Effects Inference and Tests of Correlation for Longitudinal Functional Data,” *Statistics in Medicine* 41, no. 17 (2022): 3349–3364.
16. B. Liu and H. G. Müller, “Estimating Derivatives for Samples of Sparsely Observed Functions, With Application to Online Auction

- Dynamics,” *Journal of the American Statistical Association* 104, no. 486 (2009): 704–717.
17. E. D. Ballard, D. Greenstein, P. T. Reiss, et al., “Functional Changes in Sleep-Related Arousal After Ketamine Administration in Individuals With Treatment-Resistant Depression,” *Translational Psychiatry* 14, no. 1 (2024): 238.
18. J. M. Chiou, “Dynamical Functional Prediction and Classification, With Application to Traffic Flow Prediction,” *Annals of Applied Statistics* 6, no. 4 (2012): 1588–1614.
19. Y. Khan, A. E. Ostfeld, C. M. Lochner, A. Pierre, and A. C. Arias, “Monitoring of Vital Signs With Flexible and Wearable Medical Devices,” *Advanced Materials* 28, no. 22 (2016): 4373–4395.
20. E. Cui, R. Li, C. M. Crainiceanu, and L. Xiao, “Fast Multilevel Functional Principal Component Analysis,” *Journal of Computational and Graphical Statistics* 32, no. 2 (2023): 366–377.
21. J. Goldsmith, F. Scheipl, L. Huang, et al., “Refund: Regression With Functional Data,” R Package Version 0.1–37 (2024).
22. S. Wood, “Mgcv: GAMs and Generalized Ridge Regression for R,” *R News* 1, no. 2 (2001): 20–25.
23. R Core Team, *R: A Language and Environment for Statistical Computing* (R Foundation for Statistical Computing, 2023).
24. B. Carpenter, A. Gelman, M. D. Hoffman, et al., “Stan: A Probabilistic Programming Language,” *Journal of Statistical Software* 76, no. 1 (2017): 1–32.
25. G. Wahba, “Spline Models for Observational Data,” in *CBMS-NSF Regional Conference Series in Applied Mathematics* (Society for Industrial and Applied Mathematics, 1990).
26. P. Reiss, L. Huang, and M. Mennes, “Fast Function-On-Scalar Regression With Penalized Basis Expansions,” *International Journal of Biostatistics* 6 (2010): 1.
27. C. M. Crainiceanu and A. J. Goldsmith, “Bayesian Functional Data Analysis Using WinBUGS,” *Journal of Statistical Software* 32, no. 11 (2010): 1–33.
28. D. J. Lunn, A. Thomas, N. Best, and D. Spiegelhalter, “WinBUGS—A Bayesian Modelling Framework: Concepts, Structure, and Extensibility,” *Statistics and Computing* 10 (2000): 325–337.
29. P. Eilers and B. Marx, “Flexible Smoothing With B-Splines and Penalties,” *Statistical Science* 11, no. 2 (1996): 89–121, <https://doi.org/10.1214/ss/1038425655>.
30. F. O’Sullivan, “A Statistical Perspective on Ill-Posed Inverse Problems (With Discussion),” *Statistical Science* 1, no. 4 (1986): 505–527.
31. D. Ruppert, M. Wand, and R. Carroll, *Semiparametric Regression* (Cambridge Series in Statistical and Probabilistic Mathematics Cambridge University Press, 2003).
32. S. N. Wood, N. Pya, and B. Säfken, “Smoothing Parameter and Model Selection for General Smooth Models,” *Journal of the American Statistical Association* 111, no. 516 (2016): 1548–1563.
33. P. Craven and G. Wahba, “Smoothing Noisy Data With Spline Functions,” *Numerische Mathematik* 1 (1979): 377–403.
34. G. Kimeldorf and G. Wahba, “A Correspondence Between Bayesian Estimation on Stochastic Processes and Smoothing by Splines,” *Annals of Mathematical Statistics* 41, no. 2 (1970): 495–502.
35. G. Wahba, “Bayesian “Confidence Intervals” for the Cross-Validated Smoothing Spline,” *Journal of the Royal Statistical Society: Series B: Methodological* 45, no. 1 (1983): 133–150.
36. C. Crainiceanu, D. Ruppert, and M. Wand, “Bayesian Analysis for Penalized Spline Regression Using WinBUGS,” *Journal of Statistical Software* 14, no. 14 (2005): 1–24.
37. S. Wood, *Generalized Additive Models: An Introduction With R* (Chapman and Hall/CRC, 2006).
38. S. Wood and M. S. Wood, “Package ‘mgcv,’” *R Package Version* 1, no. 29 (2015): 729.
39. F. Scheipl, L. Fahrmeir, and T. Kneib, “Spike-and-Slab Priors for Function Selection in Structured Additive Regression Models,” *Journal of the American Statistical Association* 107, no. 500 (2012): 1518–1532.
40. T. Y. Sun and D. R. Kowal, “Ultra-Efficient MCMC for Bayesian Longitudinal Functional Data Analysis,” *Journal of Computational and Graphical Statistics* (2024): 1–13.
41. J. Sartini, X. Zhou, L. Selvin, S. Zeger, and C. Crainiceanu, “Fast Bayesian Functional Principal Components Analysis,” 2025.
42. A. Leroux, C. Crainiceanu, E. Smirnova, and Q. Cao, “rnhanesdata: NHANES Accelerometry Data Pipeline,” 2019.
43. J. E. Gellar, E. Colantuoni, D. M. Needham, and C. M. Crainiceanu, “Cox Regression Models With Functional Covariates for Survival Data,” *Statistical Modelling* 15, no. 3 (2015): 256–278.
44. D. Kong, J. G. Ibrahim, E. Lee, and H. Zhu, “FLCRM: Functional Linear Cox Regression Model,” *Biometrics* 74, no. 1 (2018): 109–117.
45. S. Qu, J. L. Wang, and X. Wang, “Optimal Estimation for the Functional Cox Model,” *Annals of Statistics* 44, no. 4 (2016): 1708–1738.
46. E. Cui, C. M. Crainiceanu, and A. Leroux, “Additive Functional Cox Model,” *Journal of Computational and Graphical Statistics* 30, no. 3 (2021): 780–793.
47. M. McLean, G. Hooker, A. M. Staicu, F. Scheipl, and D. Ruppert, “Functional Generalized Additive Models,” *Journal of Computational and Graphical Statistics* 23, no. 1 (2014): 249–269.
48. S. L. Brilleman, E. M. Elci, J. B. Novik, and R. Wolfe, Bayesian Survival Analysis Using the Rstanarm R Package arXiv preprint, arXiv:2002.09633. 2020.
49. Y. J. Cheng and C. M. Crainiceanu, “Cox Models With Smooth Functional Effect of Covariates Measured With Error,” *Journal of the American Statistical Association* 104, no. 487 (2009): 1144–1154.
50. D. Cox, “Regression Models and Life-Tables,” *Journal of the Royal Statistical Society, Series B: Statistical Methodology* 34, no. 2 (1972): 187–220.
51. J. O. Ramsay, “Monotone Regression Splines in Action,” *Statistical Science* 3, no. 4 (1988): 425–441, <https://doi.org/10.1214/ss/1177012761>.
52. W. Wang and J. Yan, “Shape-Restricted Regression Splines With R Package splines2,” *Journal of Data Science* 19, no. 3 (2021): 498–517, <https://doi.org/10.6339/21-JDS1020>.
53. W. Wang and J. Yan, “splines2: Regression Spline Functions and Classes,” R Package Version 0.5.3 (2024).
54. C. M. Crainiceanu, A. M. Staicu, and C. Z. Di, “Generalized Multilevel Functional Regression,” *Journal of the American Statistical Association* 104, no. 488 (2009): 1550–1561.
55. K. Karhunen, “Under Lineare Methoden in Der Wahr Scheinlichkeitrechnung,” *Annales Academiae Scientiarum Fennicae Series A1: Mathematica Physica* 47 (1947): 1–79.
56. J. Goldsmith, V. Zipunnikov, and J. Schrack, “Generalized Multilevel Function-On-Scalar Regression and Principal Component Analysis,” *Biometrics* 71, no. 2 (2015): 344–353.
57. L. B. Mirel, L. K. Mohadjer, S. M. Dohrmann, et al., “National Health and Nutrition Examination Survey: Estimation Procedures, 2007-2010,” *Vital and Health Statistics. Series 2, Data Evaluation and Methods Research*, no. 159 (2013): 1–17.
58. A. Leroux, E. Cui, E. Smirnova, J. Muschelli, J. A. Schrack, and C. M. Crainiceanu, “NHANES 2011-2014: Objective Physical Activity Is the Strongest Predictor of All-Cause Mortality,” *Medicine and Science in Sports and Exercise* 56, no. 10 (2024): 1926–1934.

59. D. John, Q. Tang, F. Albinali, and S. Intille, “A Monitor-Independent Movement Summary to Harmonize Accelerometer Data Processing,” *Human Kinetics Journal* 2, no. 4 (2018): 268–281.
60. A. Leroux, “rnhanesdata: NHANES Accelerometry Data Pipeline,” 2019, <https://github.com/andrew-leroux/rnhanesdata>.
61. L. Xiao, V. Zippunikov, D. Ruppert, and C. Crainiceanu, “Fast Covariance Estimation for High-Dimensional Functional Data,” *Statistics and Computing* 26 (2016): 409–421.
62. L. B. Gunn-Sandell, E. J. Bedrick, J. L. Hutchins, A. A. Berg, A. M. Kaizer, and N. E. Carlson, “A Practical Guide to Adopting Bayesian Analyses in Clinical Research,” *Journal of Clinical and Translational Science* 8, no. 1 (2024): e3.

### Supporting Information

Additional supporting information can be found online in the Supporting Information section. **Data S1.** Supporting Information.