Fast Generalized Functional Principal Components Analysis

Andrew Leroux
Department of Biostatistics and Informatics University of Colorado Anschutz Medical Campus

Ciprian M. Crainiceanu
Department of Biostatistics, Johns Hopkins Bloomberg School of Public Health

Julia Wrobel
Department of Biostatistics and Informatics, University of Colorado Anschutz Medical Campus

November 2, 2022

Abstract

We propose a new fast generalized functional principal components analysis (fast-GFPCA) for dimension reduction of non-Gaussian functional data. The method consists of: (1) binning the data within the functional domain; (2) fitting local random intercept generalized linear mixed models in every bin to obtain the initial estimates of the person-specific functional linear predictors; (3) using fast functional principal component analysis to smooth the linear predictors and obtain their eigenfunctions; and (4) estimating the global model conditional on the eigenfunctions of the linear predictors. An extensive simulation study shows that fast-GFPCA performs better than existing state-of-the-art approaches, it is orders of magnitude faster, and scales up well both with the number of observed curves and observations per curve. Methods were motivated by and applied to a study of active/inactive physical activity profiles obtained from wearable accelerometers in the NHANES 2011-2014 study. The method can be implemented by any user familiar with mixed model software and could be extended to covariate adjusted and/or multilevel GFPCA.

Keywords: functional data, FPCA, others

1 Introduction

Functional data analysis (FDA) [32] provides a wide range of analysis techniques for data with complex structures such as time series or images. These data are often high dimensional (many repeated observations per function) and exhibit complex and non-stationary patterns of variation. Functional principal components analysis (FPCA) [21, 33, 36], the analog of principal components analysis (PCA) [17, 20, 30] for functional data, is a first-line dimension reduction technique for the analysis for such data. Key differences between FPCA and PCA are that functional data: (1) may be observed with substantial measurement error; (2) is expressed in the same unit of measurement at every point in the domain; and (3) points in the functional domain are typically ordered and have a natural distance between them (e.g., time ordering and difference for time series). These characteristics of the functional data can be used to improve FPCA methods.

There is a rich literature on FPCA focusing on both estimation and inference. Broadly, FPCA methods for Gaussian data involve either smoothing the covariance function [36, 48, 49] or estimation via Bayesian or likelihood approaches [37]. Extensions to sparse or irregularly observed data [18, 50, 46], multivariate [16, 7, 26] and multilevel functional [9, 8] data exist. Although many papers on FPCA have been published, there are few high quality, open source, software implementations. The covariance smoothing FACE approach of [48, 47], implemented in the `refund` package [12] in `R`, is by far the fastest approach for estimating FPCA for regularly observed data. The likelihood/Bayesian methods require substantially longer computation times for large data.

Here we focus on extensions of FPCA methods to non-Gaussian outcomes (e.g., binary or count data), which we refer to as Generalized FPCA (GFPCA). More precisely, we focus on methods that decompose the variability of the person-specific latent functional linear predictors along their main directions of variation. In contrast to the relatively large number of published papers on FPCA, there are far fewer GFPCA papers. A few exceptions are [15, 38, 11] for single-level and [6, 13] for multilevel GFPCA. Unfortunately, the software implementation problem is even more acute for GFPCA compared to FPCA. Indeed, published methods either lack accompanying open-source software or are extremely

slow for large data. Moreover, current methods require pre-specifying both the number of principal components to be estimated and the number of basis functions used to estimate the principal components. Thus, assessing sensitivity to these input parameters, key tuning parameters in FPCA like methods, requires re-estimation of the GFPCA model for each candidate input parameter value. This is a problem in the large number of applications that collect high-resolution non-Gaussian functional data. Consider, for example, the minute level physical activity data obtained from wearable accelerometers deployed in large epidemiologic studies, such as the National Health and Nutrition Survey (NHANES) 2011-2014 waves and the UK Biobank [10]. In many applications one is interested in the pattern of being active (coded as 1) or inactive (coded as 0) at every minute of the day. Thus, the data at the study participant level is a function observed at 1440 minutes (number of minutes in a day) where the value of the function is either active or inactive at every time point. NHANES contains such data for tens of thousands of study participants, while UK Biobank contains such data for close to $100,000$ study participants. Our goal is to provide GFPCA methods that extract orthogonal directions of variations in the linear predictor space of these functions.

While general purpose software for GFPCA that accommodates multiple types of exponential family data are slow, there is also good news about specific types of outcomes. For example, [45] developed a very fast and efficient algorithm for estimation of GFPCA for binary data using an EM algorithm for a variational approximation to the binomial GFPCA likelihood. The paper is accompanied by the `registr` package [42, 43]. To our knowledge this is the only publicly available GFPCA implementation that is fast and viable for large datasets, and thus we compare our fast-GFPCA approach to the `registr::bfpca()` function. In addition, for simulation scenarios with smaller data sets, we compare fast-GFPCA to the two-step GFPCA implementation described in [11], which is available in the `registr` package as the `registr::gfpca_twoStep()` function. This two-step approach can be used to model several exponential family outcomes but is prohibitively slow for large datasets. Fast GFPCA uses a different philosophy, methods and implementation from the methods implemented in the `registr` package. The advantages of fast-GFPCA are that: (1) it can

be used for any type of non-Gaussian outcome, not just binary; (2) it can easily be extended to account for covariates; and (3) it can be generalized to multilevel or longitudinal functional data. We will briefly discuss these extensions, but leave the details for future work.

The remainder of this manuscript is organized as follows. In Section 2 we present the fast GFPCA (fast-GFPCA) approach. Next, in Section 3, we apply the fast-GFPCA to active/inactive profiles obtained from wearable accelerometers using data from the National Health and Nutrition Survey (NHANES) 2011-2014 waves. We then illustrate the utility of the fast-GFPCA approach in a simulation study in Section 4. We conclude with a discussion in Section 5.

2 Methods

The observed data structure is of the type $\{s_j, Z_i(s_j)\}$, where $Z_i(s_j)$ is the observation at the point $s_j \in S$ for $j \in 1, \ldots, J$. We assume that these are discrete realizations from a continuous process $\{Z_i(s) : s \in S\}$ such that: (1) $g[E\{Z_i(s)\}] = \mu(s) + b_i(s)$, where $g(\cdot)$ is an appropriate link function, $\mu(s)$ is the population mean in the linear predictor space, and $b_i(s)$ is the individual deviation from the population mean in the linear predictor space; (2) $b_i(s) \sim \text{GP}(0, K_b)$ is a zero mean Gaussian process with covariance operator K_b. Our goals are to decompose the variability of the latent process $b_i(s)$ along its main directions of variation (obtain the PCA decomposition) and estimate $b_i(s)$ conditional on these directions of variation.

Even though $b_i(s)$ are not directly observed, we can use the Karhunen-Loève (KL) expansion $b_i(s) = \sum_{k=1}^{\infty} \xi_{ik}\phi_k(s)$ where $\phi_k : S \to \mathbb{R}$ are orthonormal eigenfunctions such that $\int_S \phi_k^2(s)ds = \lambda_k$, $\lambda_1 \geq \lambda_2 \geq \ldots$ are the eigenvalues, and $\xi_{ik} \sim N(0, \lambda_k)$ are mutually independent over study participants, i, and eigenvalues, k. This is very similar to the classical FPCA model with the exception that the noise is not necessarily Gaussian.

2.1 Fast GFPCA

We propose the following fast-GFPCA algorithm to conduct FPCA on latent processes when observed data are Gaussian: (1) bin the data along the observed functional domain $\mathcal{S} = 1, \ldots, J$ into L bins which may be overlapping; (2) estimate separate local GLMMs with a random intercept in each bin and obtain the subject-specific estimates on the linear predictor scale; (3) estimate FPCA on the subject-specific estimates obtained from step (2); (4) re-estimate GFPCA using the estimated eigenfunctions obtained from step (3) at the resolution of the original data. Each of these steps straightforward to implement, though we provide the `R` package `fastGFPCA` for convenience, described in Section 2.3. Details on each step are provided in Section 2.2 below.

2.2 Estimation Algorithm

We now provide more details on each step of the fast-GFPCA algorithm.

Step 1: The first step in the fast GFPCA algorithm is to choose how to bin the data. The choice of the number of bins (L) and the bin widths (w_l) will be informed by identifiability and assumed complexity of the underlying latent process. Specifically, suppose we choose L bins, where m_l is the midpoint of the $l = 1, \ldots, L$ bin. We use symmetric bins, except on the boundary of S. For regularly observed data the l^{th} bin contains the data $\mathcal{S}_l = \{s_{m_l - w_l/2}, \ldots, s_{m_l}, \ldots, s_{m_l + w_l/2}\}$, resulting in $w_l + 1$ points. The binned data are then $[\{Z_i(s_j), l\}, 1 \leq i \leq N, j \in \mathcal{S}_l, 1 \leq l \leq L]$. These data may be stored in list or long format.

If the data are cyclic, as in our data application, the bins may cross the boundary. For example, with minute level accelerometry data, if we let $m_1 = 1$ (activity 00:00-00:01) and $w_l = 6$, then $\mathcal{S}_1 = \{1438, 1439, 1440, 1, 2, 3, 4\}$ (activity 23:57-00:05). When the data are non-cyclic, it is not possible to construct bins in this fashion at the boundary. For boundary points with non-cyclic data we recommend constructing bins as $\mathcal{S}_l = \{\min(s_{m_l - w_l/2}, s_1), \ldots, s_{m_l}, \ldots, \max(s_{m_l + w_l/2}, s_J)\}$, resulting in bins with as few as $w_l/2 + 1$ data points.

Step 2: The next step is to fit a local GLMM in every bin. Specifically, in each bin $l = 1, \ldots, L$ we estimate separate models of the form $g[E\{Z_i(s_j)\}]| = \beta_0(s_{w_l}) + b_i(s_{w_l}) = \eta_i(s_{w_l})$ for $s_j \in \mathcal{S}_l$. Here $\beta_0(s_{w_l})$ is a fixed effect mean and $b_i(s_{w_l})$ is a random intercept evaluated at the center of the bin, s_{w_l}. From these models we obtain estimates of the global mean, $\widehat{\beta}_0(s_{w_l})$, and predictions of the subject-specific random effects $\widehat{b}_i(s_{w_l})$. These predictions are *not* on the original grid the functions were observed on, but rather the midpoints $\{s_{w_1}, \ldots, s_{w_L}\} \subset \mathcal{S}$.

Importantly, this model is, in general, misspecified because it assumes a constant effect over $s_j \in \mathcal{S}_l$ while both $\beta_0(\cdot)$ and $b_i(\cdot)$ vary smoothly over $\mathcal{S}$. This can lead to biased estimates for $b_i(s)$. However, these initial biased estimators are only used to estimate the covariance matrix and its eigenvectors in Step 3. Under mild conditions these estimators are unbiased. Indeed, if $\widehat{b}_i(s_{m_l}) = B_0(s_{m_l}) + b_i(s_{m_l}) + e_i(s_{m_l})$, where $B_0(\cdot)$ is a constant bias term over study participants, i, at every $\{s_{m_l} : 1 \leq l \leq L\}$, and $e_i(s) \sim N(0, \sigma_e^2)$ are independent random errors that are mutually independent of $b_i(\cdot)$ then

$$\widehat{\eta}_i(s_{m_l}) = \{\widehat{\beta}_0(s_{m_l}) + B_0(s_{m_l})\} + b_i(s_{m_l}) + e_i(s_{m_l}) \ ,$$

the bias term is absorbed in the intercept, and `fpca.face` function from the `refund` package [12] can be used to provide unbiased estimators of K_b using the outcome data $\widehat{\eta}_i(s)$. In addition, if the bias is constant and additive (the global mean) as well constant multiplicative on the across the domain, but instead takes the form of a constant multiple of the form

$$\widehat{\eta}_i(s_{m_l}) = \{\widehat{\beta}_0(s_{m_l}) + B_0(s_{m_l})\} + \{A_0(s_{m_l}) \times b_i(s_{m_l})\} + e_i(s_{m_l}) \ ,$$

then the correlation function is unbiased, but the variance estimates (e.g. the eigenvalues of the covariance matrix) are biased. Finally, if, in addition unbiased estimation of the correlation function, we have $\text{Var}(\tilde{b}_i(s_{m_l}))/\text{Var}(b_i(s_{m_l})) = c$, $c \in \mathbb{R}$ for all $\{s_{m_l} : 1 \leq l \leq L\}$, the eigenfunctions of the covariance matrix estimated in Step 3 are estimated in an unbiased fashion.

alternative explanation (work in progress, implicitly assumes equal, regular density of points observed along the domain): For simplicity, if we assume that the length of all windows is 1 (this simplifies normalizing constant in subsequent integrals). The misspecified

model in Step 2 estimates $\tilde{b}_i(s_{m_l}) = E[b_i(s)|\{\xi_i : 1 \leq k \leq K\}, s \in \mathcal{S}_l]$. Thus,

$$\tilde{b}_i(s_{m_l}) = E[\sum_{k=1}^{K} \phi_k(s)\xi_{ik}|\boldsymbol{\xi}_i] = \sum_{k=1}^{K} \xi_{ik} \int_{s\in\mathcal{S}_l} \phi_k(s)ds$$

resulting in subject-specific bias of $\text{Bias}(\tilde{b}_i(s_{m_l})) = \sum_{k=1}^{K} \xi_{ik} \left[\int_{s\in\mathcal{S}_l} \phi_k(s)ds - \phi_k(s_{m_l})\right]$. It follows that $\text{Var}(\tilde{b}_i(s_{m_l})) = \sum_{k=1}^{K} \left[\int_{s\in\mathcal{S}_l} \phi_k(s)ds\right]^2 \lambda_k$ and

$$\text{Cov}(\tilde{b}_i(s_{m_u}), \tilde{b}_i(s_{m_v})) = \sum_{k=1}^{K} \lambda_k \left[\int_{\mathcal{S}_u} \phi_k(s)ds\right] \left[\int_{\mathcal{S}_v} \phi_k(s)ds\right]$$

So, if $\int_{\mathcal{S}_u} \phi_k(s)ds = \phi_k(s_{m_u})$, $\int_{\mathcal{S}_v} \phi_k(s)ds = \phi_k(s_{m_v})$ for $k = 1, \ldots, K$, then we recover $\text{Cov}(\tilde{b}_i(s_{m_u}), \tilde{b}_i(s_{m_v})) = \text{Cov}(b_i(s_{m_u}), b_i(s_{m_v}))$. This exact result will hold true when, for example, all ϕ_k are constant or linear over $\mathcal{S}_u$, $\mathcal{S}_v$. When the exact result does not hold, we obtain bias of the form

$$\text{Bias}\{\text{Cov}(\tilde{b}_i(s_{m_u}), \tilde{b}_i(s_{m_v}))\} = \sum_{k=1}^{K} \lambda_k \left(\left[\int_{\mathcal{S}_u} \phi_k(s)ds\right] \left[\int_{\mathcal{S}_v} \phi_k(s)ds\right] - \phi_k(s_{m_u})\phi_k(s_{m_v}) \right)$$

This bias term tends to 0 as the length of the intervals $\mathcal{S}_u$, $\mathcal{S}_v$ tends to 0, making the fastGFPCA method particularly appealing for densely measured functional data, or data where the leading eigenfunctions of the covariance operator are not overly wiggly relative to the density of the observed data. (Andrew: should we say more here? go into limiting rates of linear approximations of functions? We can also probably quantify this exactly if we assume the covariance function lies is the space spanned by our bivariate B-spline basis (eigenfunctions are a linear combination of locally defined polynomials of degree 3))

Step 3: Use the predicted responses on the linear predictor scale $[\{\widehat{\eta}_i(s_j), l\}, 1 \leq i \leq N, j \in \mathcal{S}_l]$ to estimate $\widehat{K}_b(u, v) = \text{Cov}\{\widehat{\eta}_i(u), \widehat{\eta}_i(v)\}$ for $u, v \in \{s_{w_1}, \ldots, s_{w_L}\}$ via the fast covariance estimation (FACE) method implemented in the `refund::fpca.face()` function. Obtain the estimated eigenfunctions, $[\{\hat{\phi}_k(s_{w_l})\}, 1 \leq l \leq L, 1 \leq k \leq K]$ of the covariance operator $\widehat{K}_b$ where the number of eigenfunctions, K, is selected using, for example, the percent variance explained (e.g., 95%, 99%,).

Step 4: This step consists of estimating the GFPCA model conditional on the basis functions obtained in Step 3. Because the basis functions are estimated on a different grid from the one of the observed data, we project each eigenfunction on the rich B-spline basis used in the FACE component of the algorithm in Step 3. This provides an estimator of the eigenfunctions at every point where data was originally sampled. After these projections, fast-GFPCA becomes the following generalized linear mixed model (GLMM)

$$g(E[Z_i(s)]|\{\tilde{\phi}_k(s) : 1 \leq s \leq J, 1 \leq k \leq K\}) = \beta_0(s) + \sum_{k=1}^{K} \xi_{ik}\tilde{\phi}_k(s) \ , \tag{1}$$

where $\xi_{ik} \sim N(0, \sigma_k^2)$ are mutually independent and $\beta_0(s)$ is an unspecified smooth function. That is, given the estimates of $\tilde{\phi}_k(\cdot)$, this is a generalized linear mixed model with K uncorrelated random slopes. This is important for computational purposes as the model contains a relatively small number of variance parameters (K, because we use principal components decomposition) and the random slopes are uncorrelated (which simplifies the random effects covariance structure).

If we assume a parametric form for $\beta_0(s)$, any generalized linear mixed model software can be used. For the case when $\beta_0(s)$ is modeled nonparametrically, we estimate the model using the `mgcv::bam()` function [40] with fast REML smoothing parameter selection and the argument `discrete=TRUE` [27]. This approach is highly computation efficient; see example code in Section 2.3. Alternative software for estimating additive generalized linear mixed models are available and may be faster and more memory efficient in certain situations. Specifically, the `mgcv::gamm()` [39] and `gamm4::gamm4()` [41] functions, which provide interfaces between the `mgcv` and `nlme` [31] and `lme4` [2] packages, respectively.

The fast-GFPCA steps can be presented algorithmically as follows:

Algorithm 1: Fast-GFPCA

Step 1: Bin the Data

Input : $[\{Z_i(s_j)\}, 1 \leq i \leq N, 1 \leq j \leq J]$

Output : $[\{Z_i(s_j), l\} 1 \leq i \leq N, j \in \mathcal{S}_l, 1 \leq l \leq L\}]$

Step 2: Estimate Local GLMMs

Input : $[\{Z_i(s_j), l\} 1 \leq i \leq N, j \in \mathcal{S}_l, 1 \leq l \leq L\}]$

Output : $[\{\widehat{\eta}_i(s_{m_l}), l\}, 1 \leq i \leq N, 1 \leq l \leq L]$

for $l = 1, \ldots, L$ **do**

Fit a GLMM of the form:

$$g(E[Z_i(s_j)|s_j \in S_l]) = \beta_0(s_{m_l}) + b_i(s_{m_l}) = \eta_i(s_{m_l})$$

Obtain $\widehat{\eta}_i(s_{m_l}) = \hat{\beta}_0(s_{m_l}) + \widehat{b}_i(s_{m_l})$

end

Step 3: FPCA on Local Latent Estimates

Input : $[\{\widehat{\eta}_i(s_{m_l}), l\}, 1 \leq i \leq N, 1 \leq l \leq L]$

Output : $[\{\hat{\phi}_k(s_{m_l}), l\}, 1 \leq l \leq L, 1 \leq k \leq K]$

Estimate $\text{Cov}\{\widehat{\eta}_i(u), \widehat{\eta}_i(v)\}$ for $u, v = m_1, \ldots, m_L$ using the FACE algorithm

Obtain $\hat{\phi}_k(s)$ for $s = m_1, \ldots, m_L$

Step 4: Re-estimate Subject Specific Scores

Input : $[\{\hat{\phi}_k(s_l), l\}, 1 \leq l \leq L, 1 \leq k \leq K], [\{Z_i(s_j)\}, 1 \leq i \leq N, 1 \leq j \leq J]$

Output : $[\{\widehat{b}_i(s_j), \hat{\beta}_0(s_j)\}, 1 \leq i \leq N, 1 \leq j \leq J]$

for $k = 1, \ldots, K$ **do**

Obtain $\widehat{\phi}_k(s)$ on the original grid $s_1, \ldots, s_J$

end

Estimate GFPCA using additive generalized linear mixed models of the form

$$g(E[Z_i(s)]|\{\widehat{\phi}_k(s) : 1 \leq s \leq J, 1 \leq k \leq K\}) = \beta_0(s) + b_i(s) = \beta_0(s) + \sum_{k=1}^{K} \xi_{ik}\widehat{\phi}_k(s)$$

$$\xi_{ik} \overset{\text{iid}}{\sim} N(0, \sigma_k^2)\ ,\ \ \text{Cov}(\xi_{ik}, \xi_{il}) = 0 \ \text{ for } k \neq l$$

2.3 Fast GFPCA: Example Code

A key appeal of fast-GFPCA is that it may be implemented by anyone familiar with mixed model software. The code below illustrates how fast-GFPCA can be implemented on a

subset of the NHANES active/inactive (1/0) profiles data used in the application described in (Section 3). For illustration we use overlapping windows and equal bin width $w_l = 10$ for $l = 1, \ldots, 1440$ (both w_l and l are expressed in minutes). The general organization of the code follows the four steps of the fast-GFPCA algorithm. This code can also be run using a one-line implementation with the accompanying `fastGFPCA` package.

```
library("refund"); library("tidyverse"); library("lme4")
df      <- read.rds("NHANES_example.rds") # read in the data
J       <- length(unique(df$index))       # dimension

## Step 1: Binning decisions
bin_len <- 10                             # bin width (w)
s_m     <- 1:J                            # bin midpoints s_{m_l}

## Step 2: Fit local binned GLMMs
fit_ls  <- vector(mode="list", length=J)  # empty list to store results
for(j in s_m){                            # loop over bins
        # get indices associated with current bin S_l
        # Note: this data is cyclic, so we look across the domain
        #       using the modulo (%%) function
        sind_j <- (j-bin_len_w/2):(j+bin_len_w/2) %% 1440
        sind_j[sind_j == 0] <- 1440
        # subset to the current indices
        df_j <- df %>%
                filter(index %in% sind[sind_j])
        # fit the local, binned GLMM:
        #   g(E[Z_i(s)]|s_j \in S_l) = \beta_0(s_{m_l}) + b_i(s_{m_l})
        fit_j <- glmer(value ~ 1 + (1|id), data=df_j, family=binomial)
        # store results (id, \tilde{eta}_i(s_{m_l}), s_{m_l})
        fit_ls[[j]] <- data.frame("id" = 1:N,
                                  "eta_i" = coef(fit_j)$id[[1]],
                                  "s_m" = j)
}
## bind elements of the list row-wise
fit_df <- bind_rows(fit_ls)

## Step 3: FPCA on the binned latent estimates
fpca_latent <- fpca.face(matrix(fit_df$eta_i, N, J, byrow=FALSE),
                         pve=0.95, argvals=sind, knots=20,lower=0)

## Step 4: Re-fit the GLMM on the full data
##         Note that here we do not need to interpolate the eigenfunctions
##         as the bin midpoints contained the original observations points
```

```
# data frame of eigenfunctions (take the first four)
phi_mat <- data.frame("index" = s_m, fpca_latent$efunctions[,1:4])
colnames(phi_mat)[2:5] <- paste0("Phi", 1:4)
# merge the data
df <- df %>% left_join(phi_mat, by="index") %>% mutate(id_fac = factor(id))
# re-fit the model using mgcv::bam()
fast_GFPCA <- bam(value ~ s(index, bs="cc",k=20) +
                          s(id_fac, by=Phi1, bs="re") +
                          s(id_fac, by=Phi2, bs="re") +
                          s(id_fac, by=Phi3, bs="re") +
                          s(id_fac, by=Phi4, bs="re"),
                  data=df, family=binomial, discrete=TRUE, method="fREML")
```

For presentation simplicity, the code assumes that data are ordered by subject and then by the domain of the function. If the data are not sorted like this, slight modifications are needed in Steps 3 and 4 or data can be re-ordered before the code is run. Moreover, the subject identifier needs to be a factor variable to fit the correct model, which is why the variable `id_fac` was created in Step 4. Users unfamiliar with the *mgcv* package may be confused by the syntax in the call to `mgcv::bam()` in Step 4. The expression `s(id_fac, by=Phi1, bs="re")` constructs independent normally distributed random slopes. The same random effects specification in `lme4` would be

```
fast_GFPCA <- glmer(value ~ 1 +
                            (0+Phi1|id) + (0+Phi2|id) +
                            (0+Phi3|id) + (0+Phi4|id),
                  data=df, family=binomial)
```

However, the `lme4::glmer()` does not allow to model $\beta_0(t)$ nonparametrically and a parametric form for $\beta_0(s)$ would need to be specified. In the `lme4::glmer()` example it was assumed that $\beta_0(s) = c$, though more complex models can be used.

We consider that seeing the complete code will: (1) show how easy it is to implement the proposed methods; (2) provide a modular and modifiable platform that can be used in similar situations which require specific adjustments; and (3) support the philosophy that analytic methods are not really methods without supporting software. Indeed, it is not hard to produce code that shows how a method works in a highly controlled environment. However, it is necessary to produce software that can and is applied by many users who can understand its strengths and highlight its weaknesses in the real world of applications.

For user convenience we have developed `fastGFPCA`, an `R` package available on GitHub at `https://github.com/julia-wrobel/fastGFPCA`, which wraps the code for implementing fast-GFPCA. The detailed description of the package functionality, including examples

for both binomial and Poisson distributed functional data, is contained in the `fastGFPCA` vignette associated with the package. Briefly, the primary function in the package is `fast_gfpca()`. Function arguments `overlap` and `binwidth` allow the user to select a bin width and choose whether or not to construct overlapping windows for their data. The argument `family` is fed to the functions `lme4::glmer()` and `mgcv::bam()` and are used to specify an exponential family distribution and link function using the syntax, for example, `family = binomial(link = "logit")`. The `fast_gfpca()` function returns an object of class `"fpca"` and results can be easily visualized using the `refund.shiny` package [44].

2.4 Practical Considerations

Several practical considerations must be taken into account when applying the fast-GFPCA method. We discuss these considerations in detail below.

Identifiability. Local GLMMs may be non-identifiable or model fitting may not converge. A example is when data are binary and all (or nearly all) observations are either 0 or 1 for every study participant in a particular bin. In this case the local model is non-identifiable. Potential solutions include choosing a wider bin width, imposing a lower or upper bound on the linear predictor scale, or modifying the data. For example, [1] showed that adding two successes and two failures improves the performance of confidence intervals when estimating a probability from binary data. The idea can be used in our context by adding two successes and two failures in every bin. For our application and simulations, increasing the bin size was enough to ensure excellent performance of the methods.

Bin width. As mentioned in Section 2, the choice of bin width is an important tuning parameter in the fast GFPCA algorithm. Our simulation study presented in Section 4 illustrates this point. There is a need to balance choosing a bin size that is small enough to estimate the curvature of the latent process but not too small to make GLMM fitting unstable. A possible approach is to consider bins of increased sizes to the level where GLMMs can fit the data. The bin sizes can be increased and the stability of estimators

compared. An alternative would be to conduct smoothing at the study specific level and inspect the plots to get an idea about the complexity of the underlying functions.

Non-overlapping versus overlapping windows. The choice of whether to use non-overlapping versus overlapping windows is non-trivial, and the choice may vary with application. Non-overlapping windows reduces the number of local GLMMs that need to be estimated, though computational gains are typically minor. In contrast, overlapping windows allows estimation at a finer grid of points $s_{m_1}, \ldots, s_{m_L}$, but may induce spurious correlation is the estimated latent processes. The effects of these auto correlations, if any, are not currently known in applications, though we illustrate that the method can result in under-smoothing in our data application. A possible technical solution could be to adapting the smoothing selection criteria of FACE to allow for auto correlated data, though this exceeds the scope of our current work.

Further speeding up fast-GFPCA. The primary potential computational bottleneck for Fast-GFPCA is the need to re-estimate the subject-specific scores in Step 4 of the Fast GFPCA algorithm. We have identified three potential avenues for speeding up the current algorithm, the first of which we apply in our data application, but leave detailed investigation of each to future work. The first approach is to estimate Steps 1-3 on the entire population, obtaining population level eigenfunctions for estimating subject-specific random effects, but estimating Step 4 on randomly selected sub-samples of the data. In Step 4 all that is being done is re-estimating the population mean function and projection of the subject specific latent estimates onto the population basis estimates. For large sample sizes, such as in our data application, both of these quantities should be relatively stable in sufficiently large sub-samples (e.g. $N = 1000$).

The remaining two approaches are based on the observation in our simulations and data application that the scores obtained in Step 3 are almost perfectly correlated with those from Step 4, but are off by linear scaling factor. This scaling factor appears to be the same or similarly valued across each of the eigenfunctions. This suggests that if one were able to estimate the scaling factor, final estimates for the subject specific scores

would require less computation. We see this as achievable in at least two ways. First, one may estimate Step 4 on a subset of the data, then apply the scaling factor directly to the estimates obtained form Step 3. Alternatively, one may apply the previously suggested subset analysis to estimate the variance of subject scores, λ_k, and then treat both the estimated eigenfunctions ($[\{\hat{\phi}_k(s)\}, 1 \leq s \leq J]$) and the variance parameters as fixed. In this way, the model in Step 4 requires estimating only the population mean ($\beta_0(s)$), substantially reducing the computational complexity of Step 4. This can be done easily in the *mgcv* package, and we illustrate this idea in our data application, though we leave rigorous examination of this idea for future work.

3 Application

3.1 NHANES Accelerometry Data

The National Health and Nutrition Examination Survey (NHANES) is a large, ongoing study which provides a nationally representative sample of the non-institutionalized US population. NHANES is conducted by the Centers for Disease Control (CDC) and collects data in two-year waves with the goal of providing information on the health and nutrition of the US population. Wearable accelerometers were deployed in the 2003-2004, 2005-2006, 2011-2012, and 2013-2014 waves of NHANES.

The 2003-2006 accelerometry component of the NHANES study involved participants wearing a waist-worn accelerometer during waking hours. A guide to analyzing these data is provided in [25] and an `R` data package, `rnhanesdata` [24] is publicly available on Github at `https://github.com/andrew-leroux/rnhanesdata`. The 2011-2014 accelerometer data, released in 2021, are provided at multiple resolutions: subject, minute, and sub-second level. The subject and minute level data summarize individuals' acceleration patterns based on the new Monitor Independent Movement Summary (MIMS) unit [19]. Here, we use the 2011-2014 minute level MIMS data to construct active/inactive profiles for participants.

To obtain binary active/inactive profiles, we first threshold participants' daily MIMS data as $Y_{ij}^B(s) = 1\{Y_{ij}(s) \geq 10.558\}$, where $Y_{ij}(s)$ corresponds to the i^{th} individual's

MIMS unit on day j at minute s. We then define their active/inactive profile as $Z_i(s) = \text{median}\{Y_{ij}^B(s) : j = 1, \ldots, J_i\}$. For example, if $J_i = 7$ $Z_i(s)$ is 0 if study participant i was inactive at time s for at least 4 days and 1 otherwise. When the number of good days is even, say $J_i = 2K_i$, the median is defined as the $K_i + 1^{\text{th}}$ largest observation. The threshold for active/inactive on the MIMS unit scale was 10.558, as suggested in [22], which compared established thresholds for activity counts to MIMS units. To date, no validation studies for active/inactive MIMS thresholds have been conducted, though our methodology would apply similarly to other thresholds.

Although NHANES is a nationally representative sample, obtaining representative estimates for population quantities and/or model parameters requires the use of survey weights and/or survey design [28, 35, 23]. The intersection of survey statistics and functional data analysis is a relatively new area of research [4, 3, 5, 29] and is beyond the scope of the current work. Thus we do not account for the NHANES survey design in our data application, though it is an important direction for future methodological development.

3.2 Comparison methods and criteria

We apply the fast-GFPCA (labeled `fastGFPCA`) to the NHANES data using bin widths (w_l) of 6, 10, and 30 minutes, and both overlapping and non-overlapping intervals. We compared methods to the fast binary variational FPCA (labeled `vbFPCA`) implementation in the `registr::bfpca()` function [42, 45]. We also consider a modified version of fast-GFPCA (labeled `modified fastGFPCA`) which in Step 4 requires fitting the model (1) to sub-samples of the data (in our case four sub-samples). The approach described in [11] was not computationally feasible for the NHANES data.

For presentation simplicity and to facilitate comparisons across models, we fix the number of principal components across all methods to four. The fast-GFPCA approach estimates more than four principal components, though the additional components account for only a small proportion of the variation in the latent process.

For each approach we compare model parameters and predictive performance. Specifically, for model parameters we compare the first four estimated eigenfunctions and the

population mean function. For predictive performance, we compare the estimated in-sample log-loss associated with each model fit and the estimated area under the receiver operating curve (AUC). Finally, we compare computation times across methods. Step 2 of fast-GFPCA could be parallelized, but computation times are reported for serialized fitting of the models, which provides an upper bound for this step.

Substantial differences were identified in terms of estimated population means and eigenfunctions using the vbFPCA approach and the fast-GFPCA approach (see commentary in Section 3.3). To investigate these differences a simulation study based on the NHANES data was conducted. Specifically, we simulated data using the estimated population mean, eigenfunctions, and variance of subject-specific scores obtained from the fast-GFPCA approach using $w = 6$ and non-overlapping windows applied to the NHANES data. Results of this study are reported in Section 4.

3.3 Results

3.3.1 Data Application

Model Parameters. Figure 1 displays the estimated population means (Figure 1(a)) and first four eigenfunctions (Figure 1(b)) of the latent process. Each column of Figure 1 corresponds to a different model fit where color indicates approach (vbFPCA in red, columns 1-2, and fast-GFPCA in blue, columns 3-8). The modified fast-GFPCA approach estimates a different population mean function for each of the four randomly selected sub-samples. These sub-sample estimates of the population mean are shown as dashed lines in Figure 1(a), columns 3-8. As the modified fast-GFPCA uses the population level eigenfunctions, there are no corresponding dashed plots for the eigenfunctions.

The population mean is fairly stable across sub-samples for the modified fast-GFPCA approach (Figure 1(a), columns 3-8). We also find excellent agreement between the estimated linear predictor of the modified and unmodified GFPCA approaches; see Figure 2 and associated discussion. Moreover, the population mean and estimated eigenfunctions are similar across the various fast-GFPCA fits (overlapping vs non-overlapping windows, and bin widths), suggesting that these estimators are fairly robust to the input parameters

of the fast-GFPCA algorithm in the NHANES data.

However, there are substantial differences between the vbFPCA and fast-GFPCA approaches both in terms of the estimated population mean functions and first three eigenfunctions. The largest differences in the estimated mean functions occurs around 6AM-8AM and 9PM-12AM, where the vbFPCA approach respectively over- and under-estimates the population mean function relative to fast-GFPCA. The estimated population mean function for the vbFPCA approach suggests that, for participants with $b_i(s) = 0$, the probability of being active between 6AM-8AM is around 80-90% compared to around 50% from the fast-GFPCA approach. This result is unexpected and does not match the results in the data, where we observe approximately 50% probability of being active during this time. One potential explanation could be that the vbFPCA may provide biased estimators of the mean and that the bias may be shifted into the latent random process variation. This hypothesis appears to be validated by our data driven simulation and larger simulation study presented in Sections 4.5.3 and 4, respectively.

Linear Predictor. To understand how the differences in the estimated model parameters $(\hat{\mu}(s), \hat{\phi}_k(s))$ affect the differences in the estimated linear predictor, we estimated pairwise (model pairs) linear regressions of the linear predictor from each model fit separately by time of day. We found that there was almost perfect agreement ($R^2 > 0.99$)between the fast-GFPCA and modified fast-GFPCA estimates. Figure 2 displays the results of time-specific linear regressions of the type $E[Y] = \beta_0 + \beta_1 X$ at each minute of the day. The outcome Y is the estimated log odds obtained from fast-GFPCA with non-overlapping windows and $w = 6$ and the predictor X is the estimated log odds estimated by one of three other models: (1) fast-GFPCA estimated using non-overlapping windows with $w = 30$ (red lines); (2) vbFPCA with Kt=8 (green lines); and (3) vbFPCA with Kt=30 (blue lines). The first panel displays the estimates $\hat{\beta}_0$, the second panel displays $\hat{\beta}_1$, while the third panel displays the percent variance explained (R^2). All plots are shown as a function of time as the regressions are conducted at each time point and color coded by predictor: fast-GFPCA without overlap and window size 30 minutes (red), vbFPCA with Kt=8 (green), and vbFPCA with Kt=30 (blue).

We observe near perfect linear association between the fast-GFPCA results across the day ($R^2 \geq 0.95$, right panel), with a nearly 1-to-1 relationship (intercept ≈ 0, slope ≈ 1) during the active hours of the day ($\approx$ 10AM to 8PM). During the active hours of the day, the vbFPCA results suggest a similar 1-to-1 trend in the mean, though the lower R^2 suggests less agreement between the fast-GFPCA and vbFPCA approaches than across fast-GFPCA estimates with different input parameters.

Predictive Accuracy. The observed differences in the estimated linear predictor do not appear to translate into different predictive accuracy in terms of either AUC of log loss; see the two rightmost columns of Table 1. A possible explanation of the similar predictive accuracy but different estimation performance may be that the disagreement in predictions between models appears to occur primarily during the nighttime hours (Figure 2), when the probability of an individual being active is generally very low.

Computation Time. Computation times for each model fit are presented in the middle of Table 1 for fast-GFPCA (top rows) and the vbFPCA approach (bottom rows). For fast-GFPCA, computation times are broken down by step of the algorithm. The fast-GFPCA algorithm (top rows, "Modified" = "No") requires about 3-4 hours while vbFPCA requires 7 minutes for Kt=8 and 20 minutes for Kt=30. These computation times are driven by Step 4, which is unavoidable if the model is fit on the entire NHANES data set. Indeed, the fast-GFPCA approach here is the simplest GLMM that can be fit while maintaining the functional structure of the data (a small number of uncorrelated random effects). When Step 4 is modified (top rows, "Sped-Up" = "No") computation times decrease substantially and becomes comparable to the vbFPCA approach. From a practical perspective we have not found any substantial differences between the results obtained from the fast-GFPCA and modified fast-GFPCA approaches.

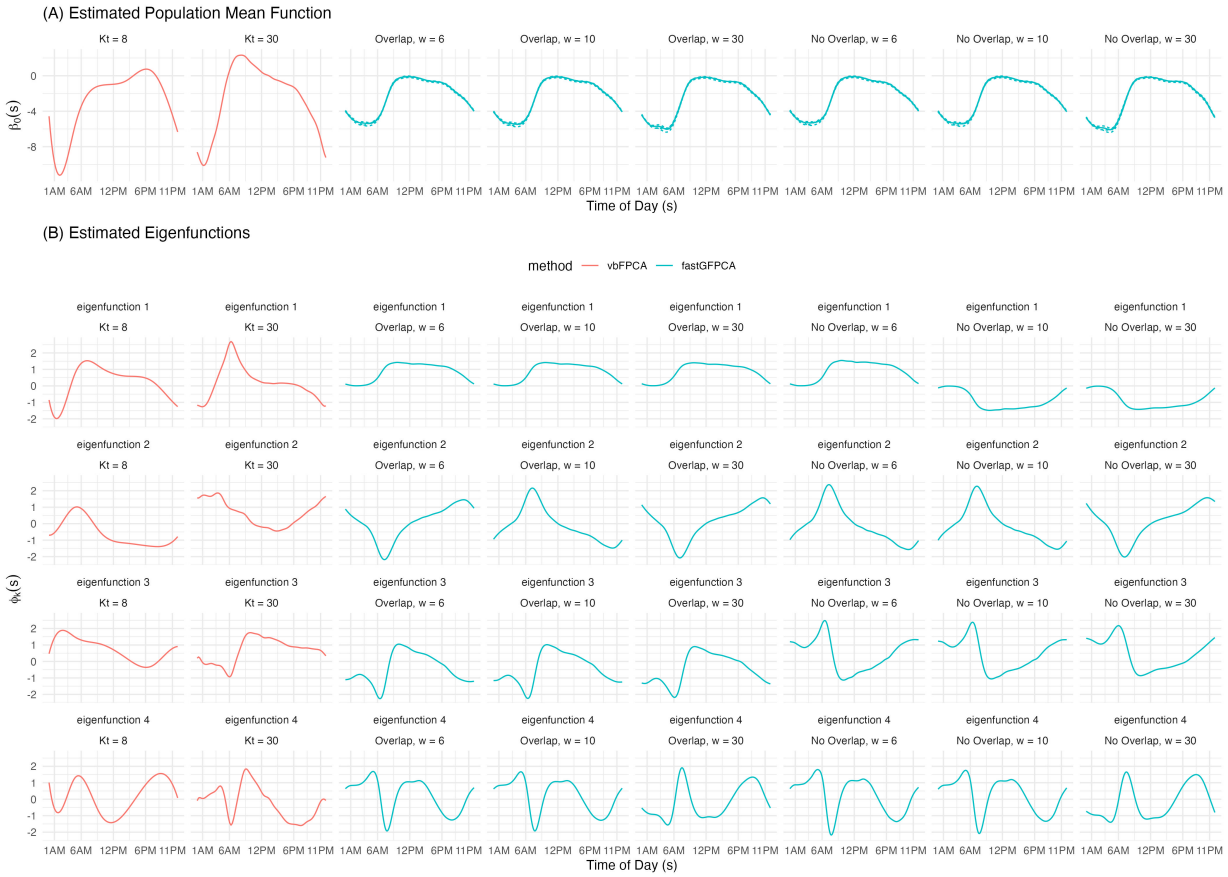

Figure 1: Estimated population mean function (first row) and the first four estimated eigenfunctions (rows $2-5$) in NHANES. Model estimates are presented as red (vbFPCA) or blue lines (fast-GFPCA). The two leftmost columns correspond to vbFPCA (Kt=8 in column 1 and Kt=30 in column 2). The six rightmost columns correspond to fast-GFPCA with different input parameters (overlapping versus non-overlapping windows) and window sizes ($w = 6, 10, 30$). Estimates of the population mean function based on the modified fast-GFPCA are displayed as dashed lines.

4 Simulation Study

Our simulations are designed to (1) quantify the computational efficiency and scalability of fast-GFPCA as sample and grid size increase, (2) evaluate the accuracy of our method in comparison with existing approaches for generalized and binary FPCA, and (3) understand the behavior of our method under different data binning strategies from Step 1 of our

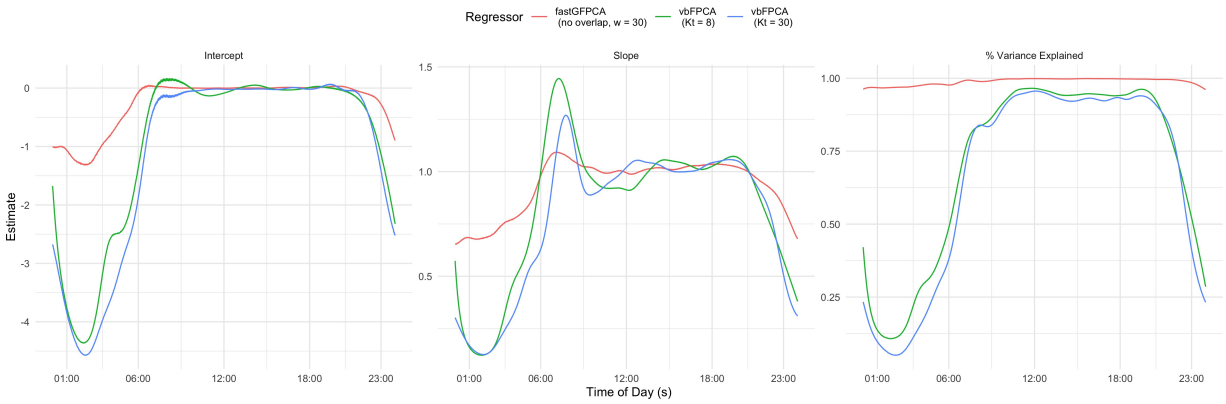

Figure 2: Results from time-specific regression of the estimated log odds of being active obtained from fast-GFPCA using $w = 6$ with non-overlapping windows and: (1) fast-GFPCA using $w = 30$ with non-overlapping windows (red lines); (2) vbFPCA with Kt = 8 (green lines); and (3) vbFPCA with Kt = 30 (blue lines). Regressions of the form $E[Y] = \beta_0 + \beta_1 X$ were fit separately for each minute, with the resulting estimates $\hat{\beta}_0$, $\hat{\beta}_1$, and the percent variance explained (R^2) plotted separately in each panel (left to right).

estimation algorithm in Section 2.2. For larger simulated datasets we focus on the binary functional data setting because the only existing competing approach that is computationally feasible for large functional datasets is tailored specifically for binary data. For smaller datasets we also evaluate our method in comparison to an existing approach for FPCA of Poisson functional data.

4.1 Simulation Design

We simulate binary and Poisson functional data on a length J grid in $S = [0, 1]$ that is equally spaced and shared across subjects. Curves for $i \in 1, \ldots, N$ subjects are drawn from the model

$$g(E[Z_i(s)]|\{\phi_k(s) : 1 \leq s \leq J, 1 \leq k \leq 4\}) = \eta_i(s)$$
$$= \mu(s) + \sum_{k=1}^{4} \xi_{ik}\phi_k(s) \ ,$$

Fast-GFPCA								
Parameters	Sped-Up	Computation Time (mins)					AUC	Log-loss
		Step 1	Step 2	Step 3	Step 4	Total		
Overlap, $w = 6$	No		16.42	0.04	174.20	190.63	0.909	0.328
	Yes		16.42	0.04	10.90	27.33	0.909	0.328
Overlap, $w = 10$	No		23.28	0.04	184.82	208.12	0.909	0.327
	Yes		23.28	0.04	10.74	34.03	0.909	0.327
Overlap, $w = 30$	No		55.90	0.04	184.66	240.57	0.909	0.327
	Yes		55.90	0.04	10.48	66.38	0.909	0.327
No Overlap, $w = 6$	No		2.48	0.01	188.90	191.43	0.909	0.328
	Yes		2.48	0.01	11.34	13.13	0.909	0.328
No Overlap, $w = 10$	No		2.19	0.01	185.04	187.28	0.909	0.327
	Yes		2.19	0.01	10.89	13.13	0.909	0.327
No Overlap, $w = 30$	No		1.97	0.01	197.35	199.36	0.909	0.327
	Yes		1.97	0.01	11.60	13.61	0.909	0.327
vbFPCA (`registr::bfpca()`)								
Parameters		Computation Time (mins)					AUC	Log-loss
Kt=8						7.11	0.909	0.328
Kt=30						20.86	0.910	0.326

Table 1: Table showing computational times for various NHANES fits

where $\mu(s)$ is a population-level mean function, $\xi_{ik} \sim N(0, \lambda_k)$ are subject-specific scores, and $\boldsymbol{\phi}(s)$ are population-level eigenfunctions. We assume $K = 4$ principal components, with true eigenvalues $\lambda_k = 0.5^{k-1}, k = 1, 2, 3, 4$. The true eigenfunctions are drawn from one of two scenarios intended to mimic real data settings. In the first setting, latent curves $\eta_i(s)$ and eigenfunctions are periodic, with $\boldsymbol{\phi}(s) = \{\sqrt{2}sin(2\pi s), \sqrt{2}cos(2\pi s), \sqrt{2}sin(4\pi s), \sqrt{2}cos(4\pi s)\}$. The second setting does not exhibit periodicity, with true eigenfunctions given by $\boldsymbol{\phi}(s) = \{1, \sqrt{3}(2s-1), \sqrt{5}(6s^2 - 6s + 1), \sqrt{7}(20s^3 - 30s^2 + 12s - 1)\}$. For most simulation settings we assume $\mu(s) = 0$, however, for a subset of scenarios we construct nonzero $\mu(s)$ using a B-spline basis.

For binary and Poisson data, $g(\cdot)$ is taken to be the the logit link and log link, respectively. For each subject and timepoint, exponential family observations $Z_i(s)$ are sampled independently from either a Bernoulli distribution with probability $logit^{-1}[\eta_i(s)]$, or from a Poisson distribution with rate $e^{[\eta_i(s)]}$.

4.2 Comparison to Existing Approaches

4.2.1 Binary Functional Data

In the binary setting we evaluate the performance of our fast-GFPCA algorithm as a function of sample sizes $N \in (100, 500, 1000)$, grid lengths $J \in (100, 500, 2000)$, and the two periodicity scenarios with $\mu(s) = 0$, and simulate 100 datasets for each combination of simulation parameters, resulting in 18 simulation scenarios. To quantify the bias in $\mu(s)$ and $\boldsymbol{\phi}(s)$ in the variational EM approach discussed in Section 4.5.3 we also employ two nonzero $\mu(s)$ simulation scenarios, with (1) $N, J = (1000, 2000)$ and (2) $N, J = (500, 100)$. In addition, we assess the performance of fast-GFPCA across different bin widths w_l and compare non-overlapping and overlapping windows. Specifically, for each simulation scenario we evaluate three different bin widths, $w_l \in (5, 10, 50)$. For periodic simulated data we consider both non-overlapping and overlapping windows across the three bin widths, and for non-periodic data we only allow non-overlapping windows for our algorithm.

To provide a frame of reference we compare fast-GFPCA with two different binary FPCA approaches, both of which are implemented in the `registr` package [42, 43]. The first method to which we compare is the two-step conditional GFCPA model introduced by [11], which is implemented using the `registr::gfpca_twoStep()` function and we refer to as *tsGFPCA* in text and figures below. While `gfpca_twoStep()` is a general purpose function that can accommodate multiple exponential family distributions, it is very computationally intensive and thus impractical for our simulation and real data settings. To reduce this computational burden we only implement *tsGFPCA* simulation settings where $N = 500$ and $J = 100$.

We compare fast-GFPCA in all simulation settings to the vbFPCA algorithm from [45], which is implemented via the `registr::bfpca()` function, and is denoted *vbFPCA*.

This method is highly computationally efficient, but designed for binary functional data with a `logit` link, and cannot be generalized to other link functions or exponential family distributions. Because the *vbFPCA* approach models population mean $\mu(s)$ and eigenfunctions $\boldsymbol{\phi(s)}$ using a B-spline expansion without a smoothness penalty, the number of basis functions must be manually tuned to obtain optimal smoothness. To address this, for each simulated dataset we implement *vbFPCA* with 8 basis functions (the package default) and 30 basis functions.

For both competing methods we implement a periodic B-spline basis using the `registr` option `periodic = TRUE` when simulated data are periodic. By default in the `registr` package, eigenfunctions $\boldsymbol{\phi(s)}$ are returned unscaled and on a grid of size 100. To enable comparison with results from fast-GFPCA, we linearly interpolate eigenfunctions estimated using *tsGFPCA* and *vbFPCA* to a grid of size J and scaled by the square root of the grid length.

4.2.2 Poisson Functional Data

In the Poisson setting we compare fast-GFPCA with `registr::gfpca_twoStep()`. Since the comparative method is highly computationally intensive, we only consider small data settings of sample sizes $N \in (50, 100)$, grid lengths $J \in (100, 200)$, and restrict our comparison to periodic data. We simulate 100 datasets for each of the six simulation scenarios arising from this combination of grid length and sample size. For fast-GFPCA we compare non-overlapping and overlapping windows, and consider bin widths $w_l \in (5, 10)$.

4.3 Evaluation Criteria

We compare the performance of the three methods (*tsGFPCA*, *vbFPCA*, and *fastGFPCA*) with respect to accuracy in recovering population-level eigenfunctions $\boldsymbol{\phi(s)}$, accuracy in recovering subject-specific latent means in the linear predictor space ($\eta_i(s)$), and computational efficiency. Accuracy of eigenfunction estimation was quantified using mean integrated squared error (MISE), defined by $\frac{1}{k}\sum_{k=1}^{4}\int_0^1 \left(\hat{\phi}_k(s) - \phi_k(s)\right)^2 ds$. Accuracy of subject-specific log-odds across models are compared using MISE across subjects, given by

$\frac{1}{N}\sum_{i-1}^{N}\int_0^1 \left(\hat{\eta}_i(s) - \eta_i(s)\right)^2 ds$. Computation times are reported in minutes. We also report computation times for steps 2 and 4 of *fastGFPCA*.

4.4 Simulation Results: Accuracy

4.5 Simulation Results: Computational Efficiency

4.5.1 Binary FPCA

4.5.2 Poisson FPCA

4.5.3 Data Driven Simulation

To investigate the differences in estimating the mean and principal components between vbFPCA and fast-GFPCA approaches we conducted a short simulation study. First the parameters of model (1) were estimated for $K = 4$ eigenfunctions using the non-overlapping fast-GFPCA model using a window size of $w = 6$. We generated five data sets with $N = 4000$ participants, each observed on the same grid as the original data ($J = 1440$). Since we did not find parameter estimates differed in the fast-GFPCA approach by bin width and overlapping versus non-overlapping windows, we only estimate the fast-GFPCA model with $w = 6$ and non-overlapping windows.

Figure 5 plots the results of this small simulation study and is structured the same as Figure 1, with the estimated population mean function $\hat{\mu}(s)$ and eigenfunctions $\hat{\phi}_k(s)$ plotted in the first and second through fifth rows, respectively. The true mean and eigenfunctions are plotted as solid black lines, while the estimated values across the 5 simulated datasets are plotted as semi-transparent red (vbFPCA) and blue (fast-GFPCA) models, respectively. We see that both the population mean function and eigenfunction estimates are biased in the vbFPCA approach and the bias aligns remarkably well with the estimates we obtained from the real data. In contrast, fast-GFPCA estimates show minimal bias. As expected with the relatively large sample size $N = 4000$), estimates of both the population mean and eigenfunctions are extremely consistent across simulated datasets. These findings suggest that the vbFPCA approach may indeed be providing biased parameter estimates

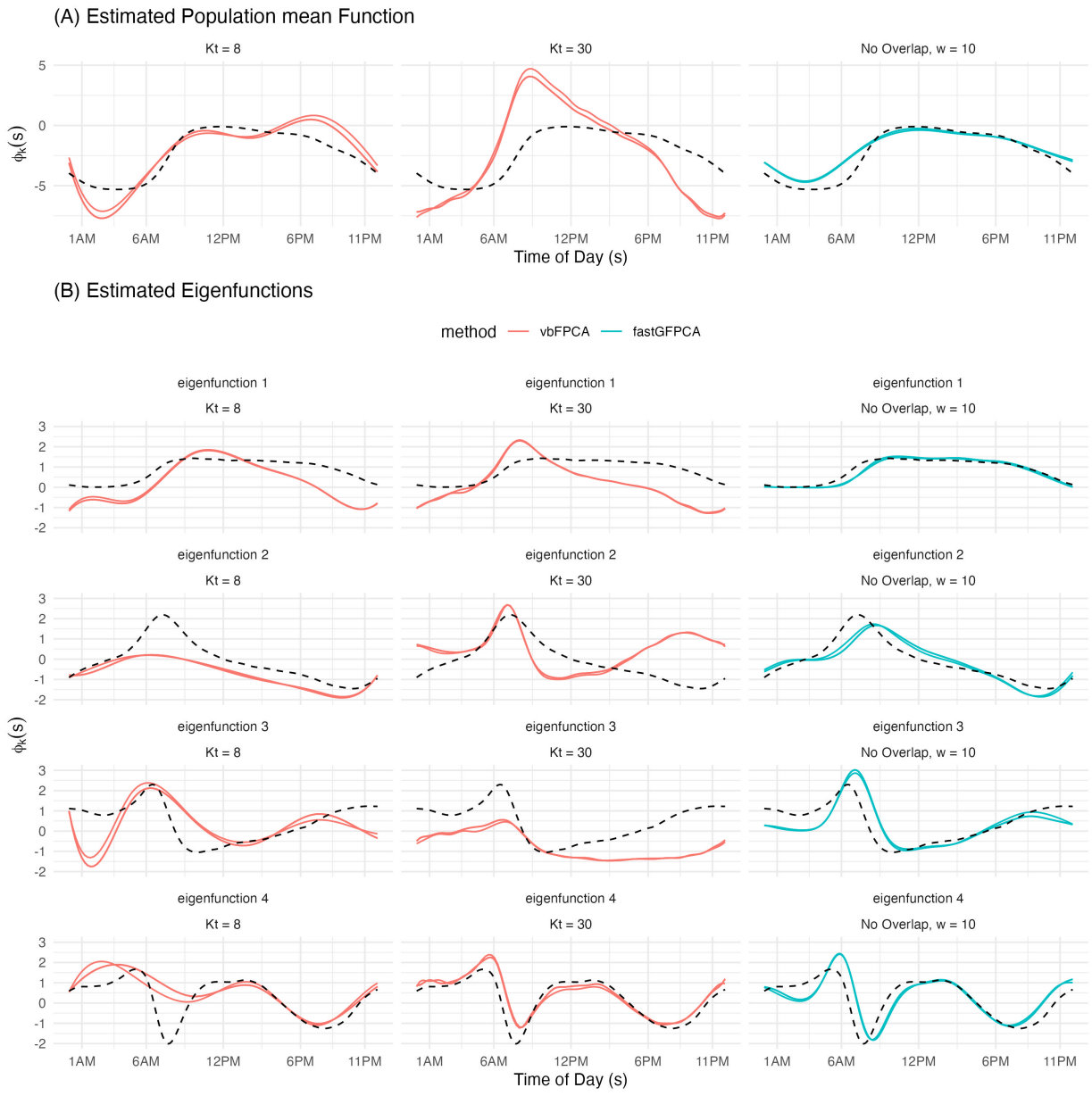

Figure 3: Figure showing the estimated population mean function ($\hat{\mu}(s)$) and the first four estimated eigenfunctions ($\hat{\phi}_k(s)$) from 5 simulated datasets based on the NHANES fast-GFPCA results. Model estimates are presented as red lines (vbFPCA) or blue lines (fast-GFPCA). vbFPCA estimates from models using either Kt=8 (left column) or Kt=30 (middle column) basis functions are presented and compared with fast-GFPCA estimated using non-overlapping widows of size $w = 10$.

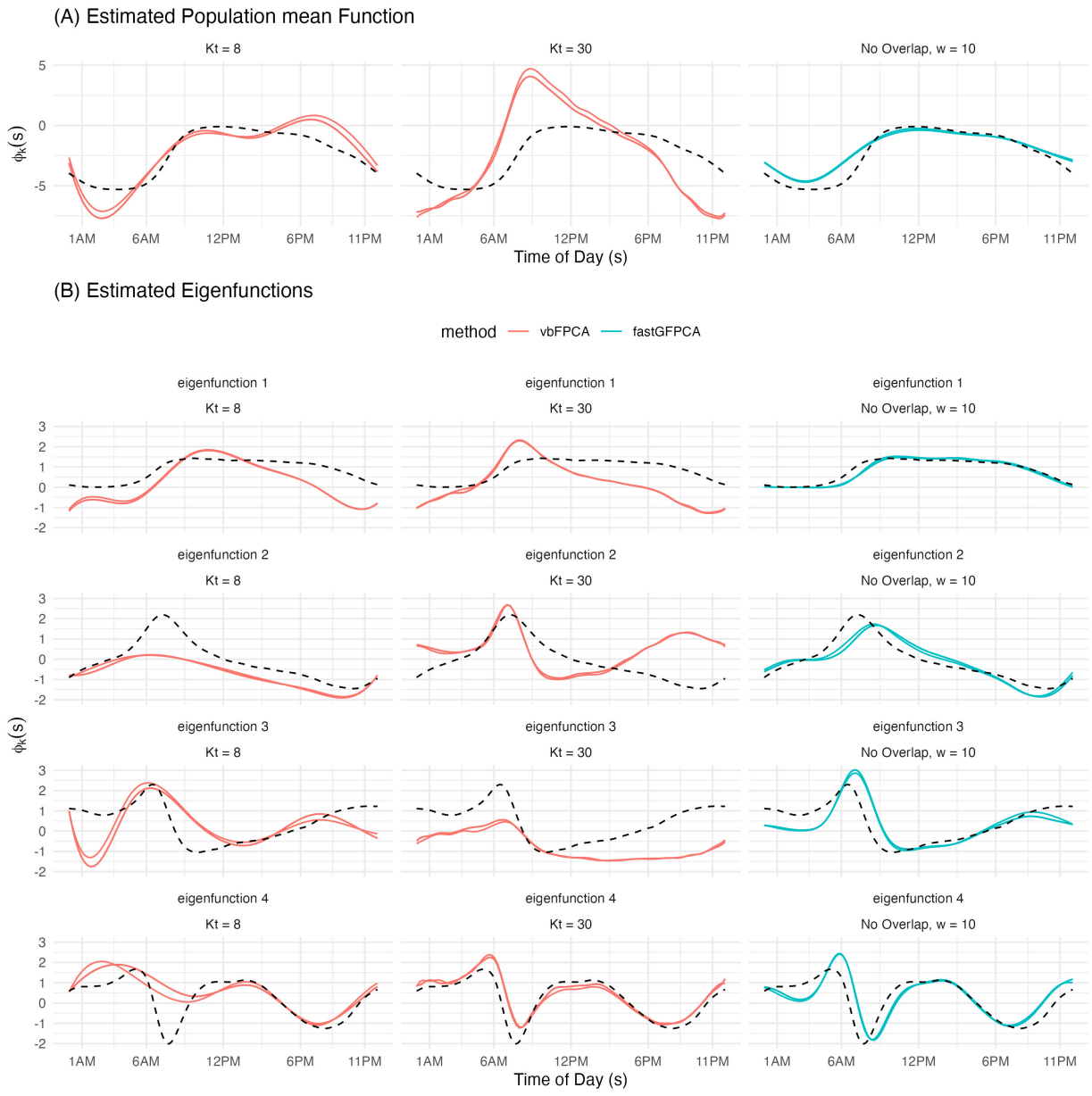

Figure 4: For binary data in words here say what the results were for tsGFPCA, since we are not including them

in our data application.

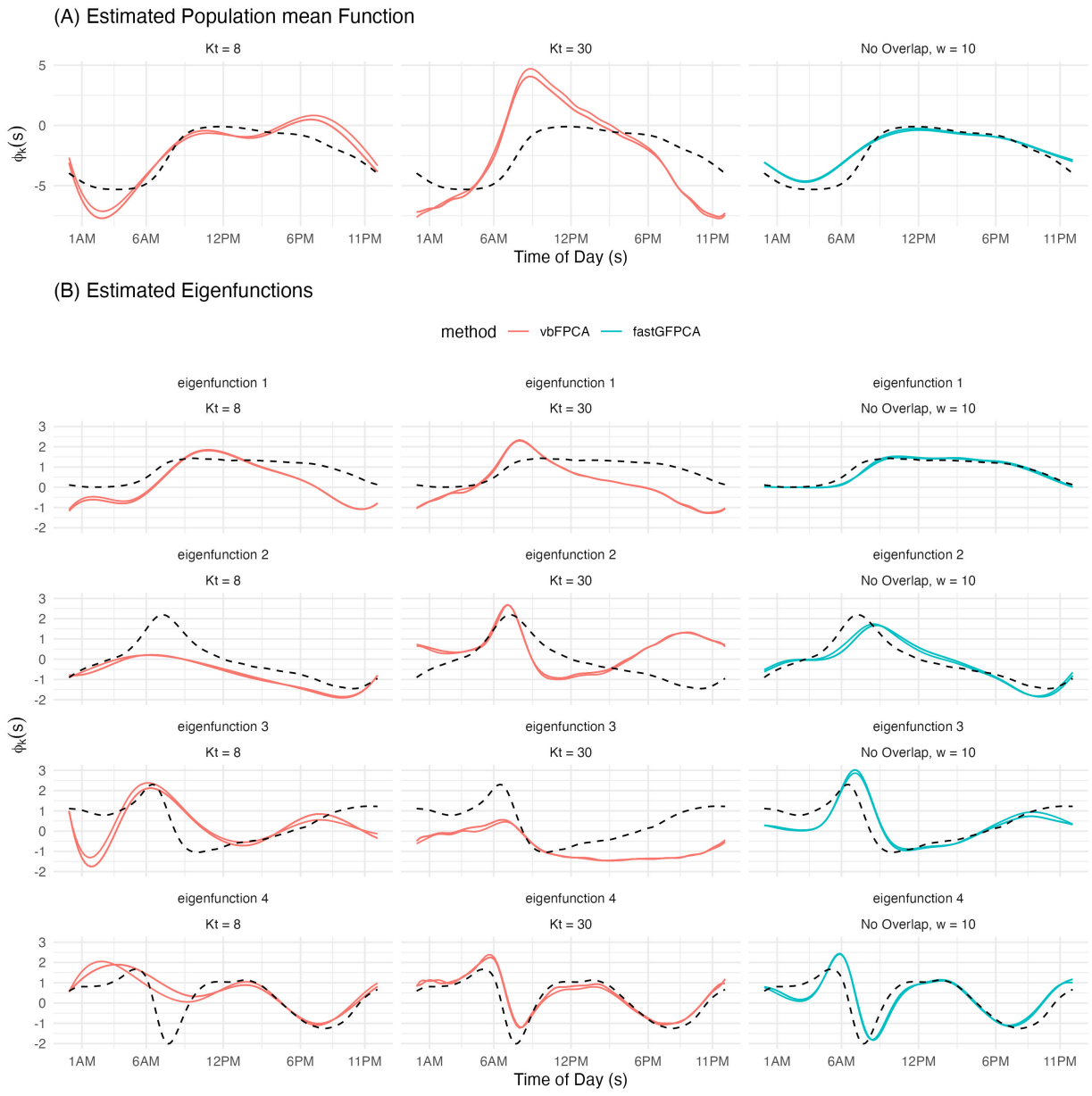

Figure 5: Figure showing the estimated population mean function ($\hat{\mu}(s)$) and the first four estimated eigenfunctions ($\hat{\phi}_k(s)$) from 5 simulated datasets based on the NHANES fast-GFPCA results. Model estimates are presented as red lines (vbFPCA) or blue lines (fast-GFPCA). vbFPCA estimates from models using either Kt=8 (left column) or Kt=30 (middle column) basis functions are presented and compared with fast-GFPCA estimated using non-overlapping widows of size $w = 10$.

5 Discussion

The fast-GFPCA method proposed in this manuscript represents a simple, understandable, and feasible solution to the relative computational complexity of estimation of functional principal components analysis for non-Gaussian data. In addition, we have provided a mathematical justification for the principals which underlie fast-GFPCA and shown that the method compares extremely favorably to the very limited existing approaches in terms of both estimation accuracy and computational efficiency. Moreover, existing methods, such as the vbFPCA approach for binary data presented here may provide biased estimates of model parameters (population mean function and covariance operator), suggesting that the fast-GFPCA approach is a reasonable method for comparison even when an alternative approach is desired in a given application.

Though the work here shows the Fast-GFPCA to be fast, accurate, and appropriate in certain contexts, methodologic work remains. Specifically, it is unclear at this time how to choose the optimal bin selection procedure, both with regard to bin width and the decision to use overlapping versus non-overlapping windows for estimation. While a cross-validated prediction error criteria may be a viable option, subject-level cross-validation requires prediction of random effects in non-Gaussian models using participants' data not included in model fitting, a non-trivial problem in non-Bayesian contexts. Moreover, when using non-overlapping windows, automated smoothing parameter selection of FPCA on the latent process in Step 3 of the Fast-GFPCA algorithm is unreliable. Here we propose an ad-hoc solution based on visual inspection of the eigenfunctions and/or estimated covariance function. While this is feasible due to the incredible speed of the FACE method, we would prefer a fully automated approach. Deriving an appropriate variation of the GCV criteria used by FACE for smoothing parameter selection which accounts for autocorrelated data would improve the method proposed here.

Nevertheless, the results of this work represent an encouraging step forward for estimation of GFPCA in very high dimensional data, specifically large N, a key bottleneck for the application of functional data analysis methods in practice. Moreover, the work here suggests relatively straightforward extensions to more complex models, including covariate

dependent GFPCA, multilevel GFPCA, and other such related models.

5.1 Extensions

An appealing feature of fast-GFPCA is that it can be extended to at least three important contexts: (1) covariate dependent GFPCA; (2) multilevel GFPCA; and (3) sparse and/or irregularly observed functional data. We briefly discuss how the Fast GFPCA method presented here may be extended to these scenarios, but leave details for future work.

Covariate dependent GFPCA. The fast-GFPCA method easily incorporates covariates into the model. Consider the case of one additional scalar predictor (e.g., age), denoted x_i. Step 2 of fast-GFPCA is simply modified to fit local models of the form $g(E[Z_i(s_j)|s_j \in S_l]) = \beta_0(s_{m_l}) + \beta_1(s_{m_l})x_i + b_i(s_{m_l}) = \eta_i(s_{m_l})$. Then, in Step 4, the final GLMM includes the additive term associated with the proposed varying coefficient model. The effort required for this extension is minimal.

Nested, longitudinal or crossed design GFPCA. Consider the case when multiple functions are observed per study participant. For example, in the NHANES data each study participant has multiple days of accelerometry data. For notation simplicity assume that there are K functions for every study participant. The extension to multilevel GFPCA follows naturally from the fast-GFPCA algorithm. Specifically, in Step 2 we fit the multilevel model $g(E[Z_{ik}(s_j)]) = \beta_0(s_{m_l}) + b_i(s_{m_l}) + v_{ik}(s_{m_l})$. Step 3 can then proceed with MFPCA FACE [8] to estimate the principal directions of variation at each level. If the functional data has longitudinal [14] or crossed [34] designs the local GLMM can be changed accordingly.

Sparse and/or irregularly observed functional data. The fast-GFPCA approach using local GLMM fitting is, in principle, extendable to sparse and irregularly observed data. However, this will require methods that account for: (1) unequal number of observations per subject and bin; and (2) unequally sampled data within a bin. This is a

very promising area of future research, though for now we focus on dense equally sampled non-Gaussian functional data.

References

[1] Agresti, A. and Caffo, B. (2000). Simple and effective confidence intervals for proportions and differences of proportions result from adding two successes and two failures. *The American Statistician*, 54(4):280–288.

[2] Bates, D., Mächler, M., Bolker, B., and Walker, S. (2015). Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1):1–48.

[3] Cardot, H., Dessertaine, A., Goga, C., Josserand, É., and Lardin, P. (2013a). Comparison of different sample designs and construction of confidence bands to estimate the mean of functional data: An illustration on electricity consumption. *Survey Methodology*, 39(2):283–301.

[4] Cardot, H., Goga, C., and Lardin, P. (2013b). Uniform convergence and asymptotic confidence bands for model-assisted estimators of the mean of sampled functional data. *Electronic journal of statistics*, 7:562–596.

[5] Cardot, H., Goga, C., and Lardin, P. (2014). Variance estimation and asymptotic confidence bands for the mean estimator of sampled functional data with high entropy unequal probability sampling designs. *Scandinavian Journal of Statistics*, 41(2):516–534.

[6] Chen, H., Wang, Y., Paik, M. C., and Choi, H. A. (2013). A marginal approach to reduced-rank penalized spline smoothing with application to multilevel functional data. *Journal of the American Statistical Association*, 108(504):1216–1229. PMID: 24497670.

[7] Chiou, J.-M., Chen, Y.-T., and Yang, Y.-F. (2014). Multivariate functional principal component analysis: A normalization approach. *Statistica Sinica*, 24(4):1571–1596.

[8] Cui, E., Li, R., Crainiceanu, C. M., and Xiao, L. (2022). Fast multilevel functional principal component analysis. *Journal of Computational and Graphical Statistics*, 0(ja):1–33.

[9] Di, C., Crainiceanu, C., Caffo, B., and Punjabi, N. (2009). Multilevel functional principal component analysis. *Annals of Applied Statistics*, 3(1):458–488.

[10] Doherty, A., Jackson, D., Hammerla, N., Plötz, T., Olivier, P., Granat, M. H., White, T., van Hees, V. T., Trenell, M. I., Owen, C. G., Preece, S. J., Gillions, R., Sheard, S., Peakman, T., Brage, S., and Wareham, N. J. (2017). Large scale population assessment of physical activity using wrist worn accelerometers: The uk biobank study. *PLOS ONE*, 12(2):1–14.

[11] Gertheiss, J., Goldsmith, J., and Staicu, A.-M. (2017). A note on modeling sparse exponential-family functional response curves. *Computational Statistics & Data Analysis*, 105:46–52.

[12] Goldsmith, J., Scheipl, F., Huang, L., Wrobel, J., Di, C., Gellar, J., Harezlak, J., McLean, M., Swihart, B., Xiao, L., Crainiceanu, C., and Reiss, P. (2020). *refund: Regression with Functional Data*. R package version 0.1-23.

[13] Goldsmith, J., Zipunnikov, V., and Schrack, J. (2015). Generalized multilevel function-on-scalar regression and principal component analysis. *Biometrics*, 71(2):344–353.

[14] Greven, S., Crainiceanu, C., Caffo, B., and Reich, D. (2010). Longitudinal functional principal component analysis. *Electronic Journal of Statistics*, pages 1022–1054.

[15] Hall, P., Müller, H.-G., and Yao, F. (2008). Modelling sparse generalized longitudinal observations with latent gaussian processes. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(4):703–723.

[16] Happ, C. and Greven, S. (2015). Multivariate functional principal component analysis for data observed on different (dimensional) domains. *Journal of the American Statistical Association*, 113:649 – 659.

[17] Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:498–520.

[18] J., P. and Paul, D. (2009). A geometric approach to maximum likelihood estimation of the functional principal components from sparse longitudinal data. *Journal of Computational and Graphical Statistics*, 18(4):995–1015.

[19] John, D., Tang, Q., Albinali, F., and Intille, S. (2019). An open-source monitor-independent movement summary for accelerometer data processing. *Journal for the Measurement of Physical Behaviour*, 2(4):268 – 281.

[20] Jolliffe, I. (1982). A note on the use of principal components in regression. *Journal of the Royal Statistical Society, Series C*, 31(3):300–303.

[21] Jones, M. C. and Rice, J. A. (1992). Displaying the important features of large collections of similar curves. *The American Statistician*, 46(2):140–145.

[22] Karas, M., Muschelli, J., Leroux, A., Urbanek, J. K., Wanigatunga, A. A., Bai, J., Crainiceanu, C. M., and Schrack, J. A. (2022). Comparison of accelerometry-based measures of physical activity: Retrospective observational data analysis study. *JMIR Mhealth Uhealth*, 10(7):e38077.

[23] Korn, E. L. and Graubard, B. I. (2011). *Analysis of health surveys*, volume 323. John Wiley & Sons.

[24] Leroux, A. (2022). *rnhanesdata: NHANES Accelerometry Data Pipeline*. R package version 1.02.

[25] Leroux, A., Di, J., Smirnova, E., McGuffey, E. J., Cao, Q., Bayatmokhtari, E., Tabacu, L., Zipunnikov, V., Urbanek, J. K., and Crainiceanu, C. (2019). Organizing and analyzing the activity data in nhanes. *Statistics in Biosciences*, 11(2):262–287.

[26] Li, C. and Xiao, L. (2021). *mfaces: Fast Covariance Estimation for Multivariate Sparse Functional Data*. R package version 0.1-3.

[27] Li, Z. and Wood, S. N. (2020). Faster model matrix crossproducts for large generalized linear models with discretized covariates. *Statistics and Computing*, 30(1):19–25.

[28] Lumley, T. (2004). Analysis of complex survey samples. *Journal of Statistical Software*, 9(1):1–19.

[29] Parker, P. A. and Holan, S. H. (2022). A bayesian functional data model for surveys collected under informative sampling with application to mortality estimation using nhanes. *Biometrics*, n/a(n/a).

[30] Pearson, K. (1901). LIII. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572.

[31] Pinheiro, J. C. and Bates, D. M. (2000). *Mixed-Effects Models in S and S-PLUS*. Springer, New York.

[32] Ramsay, J. and Silverman, B. (2005). *Functional Data Analysis*. Springer New York, NY, USA.

[33] Rice, J. and Silverman, B. (1991). Estimating the mean and covariance structure nonparametrically when the data are curves. *Journal of the Royal Statistical Society. Series B (Methodological)*, 53(1):233–243.

[34] Shou, H., Zipunnikov, V., Crainiceanu, C., and Greven, S. (2015). Structured functional principal component analysis. *Biometrics*, 71(1):247–257.

[35] Skinner, C., Wakefield, J., et al. (2017). Introduction to the design and analysis of complex survey data. *Statistical Science*, 32(2):165–175.

[36] Staniswalis, J. and Lee, J. (1998). Nonparametric regression analysis of longitudinal data. *Journal of the American Statistical Association*, 93(444):1403–1418.

[37] van der Linde, A. (2008). Variational Bayesian functional PCA. *Computational Statistics & Data Analysis*, 53(2):517–533.

[38] van der Linde, A. (2009). A Bayesian latent variable approach to functional principal components analysis with binary and count data. *AStA Advances in Statistical Analysis*, 93(3):307–333.

[39] Wood, S. N. (2011). Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(1):3–36.

[40] Wood, S. N., Li, Z., Shaddick, G., and Augustin, N. H. (2017). Generalized additive models for gigadata: Modeling the u.k. black smoke network daily data. *Journal of the American Statistical Association*, 112(519):1199–1210.

[41] Wood, S. N., Scheipl, F., and Faraway, J. J. (2013). Straightforward intermediate rank tensor product smoothing in mixed models. *Statistics and Computing*, 23(3):341–360.

[42] Wrobel, J. (2018). register: Registration for exponential family functional data. *Journal of Open Source Software*, 3(22):557.

[43] Wrobel, J. and Bauer, A. (2021). registr 2.0: Incomplete curve registration for exponential family functional data. *Journal of Open Source Software*, 6(61):2964.

[44] Wrobel, J., Park, S. Y., Staicu, A. M., and Goldsmith, J. (2016). Interactive graphics for functional data analyses. *Stat*, 5(1):108–118.

[45] Wrobel, J., Zipunnikov, V., Schrack, J., and Goldsmith, J. (2019). Registration for exponential family functional data. *Biometrics*, 75(1):48–57.

[46] Xiao, L., Li, C., Checkley, W., and Crainiceanu, C. (2018). Fast covariance estimation for sparse functional data. *Statistics and Computing*, 28(3):511–522.

[47] Xiao, L., Li, C., Checkley, W., and Crainiceanu, C. (2021). *face: Fast Covariance Estimation for Sparse Functional Data*. R package version 0.1-6.

[48] Xiao, L., Zipunnikov, V., Ruppert, D., and Crainiceanu, C. (2016). Fast covariance estimation for high-dimensional functional data. *Statistics and computing*, 26(1):409–421.

[49] Yao, F., Müller, H., Clifford, A., Dueker, S., Follett, J., Lin, Y., Buchholz, B., and Vogel, J. (2003). Shrinkage estimation for functional principal component scores with application to the population kinetics of plasma folate. *Biometrics*, 59(3):676–685.

[50] Yao, F., Müller, H.-G., and Wang, J.-L. (2005). Functional data analysis for sparse longitudinal data. *Journal of the American Statistical Association*, 28(100):577–590.

Supplemental Material