FAST GENERALIZED FUNCTAIONAL ON SCALAR REGRESSION

by  
DUSTIN MATTHEW ROGERS

B.S., Drake University 2013

A thesis submitted to the  
Faculty of the Graduate School of the  
University of Colorado in partial fulfillment  
of the requirements for the degree of  
Master of Science  
Biostatistics Program  
2023

This thesis for the Master of Science degree by

Dustin Matthew Rogers

Has been approved for the

Biostatistics Program

by

Andrew Leroux, Chair  
Julia Wrobel  
Nichole Carlson

Date: May 19th, 2023

Rogers, Dustin Matthew (MS, Biostatistics)

fast Generalized Functional on Scalar Regression

Thesis directed by Assistant Professor Andrew Leroux

ABSTRACT

Statistical methods for regression models with functional data are an area of active methodologic and computational research. Functional data are generated by random variables which are characterized by a continuous (latent) process observed at a discrete set of points. These data present computational challenges due to the fact that: 1) functional data are often high dimensional; and 2) the latent processes are often highly non-linear. The computational challenges are particularly acute when the functional data are non-Gaussian. Under this modelling framework, existing methods are not viable for moderate to large sample sizes (e.g. N ≥ 500 in many cases). The current work proposes and evaluates the feasibility and accuracy of the fast generalized functional on scalar regression (fGFoSR) method for regressing non-Gaussian functional outcomes on scalar covariates, allowing for complex non-linear associations between covariates and response, as well as highly non-linear subject-specific random effects.

Currently, general purpose methods and software for estimating this class of models is limited, with a notable exception of the penalized flexibile functional regression methodology for exponential family data implemented in the refund package in the statistical software R via the refund::pffr() function. We show that the fGFoSR method provides comparable or better estimation and inference for fixed effects when sample sizes are small to moderate (N ≤ 500). However, for large sample sizes, existing methods are not computationally feasible, with computation times exceeding 24 hours. In contrast, the fGFoSR method provides accurate estimation and inference for large sample sizes, with substantially improved computation times. We illustrate the fGFoSR method with a simulation study and data application involving binary functional data.

Approved: Andrew Leroux

**TAB LE OF CONTENTS**

CHAPTER

1. INTRODUCTION
2. METHODS

II.1 Fast Generalized Functional on Scalar Regression Implementation

II.2 Simulation Study

II.2.1 Simulation Design

II.2.2 Comparison to Existing Approaches

II.3 Application to Real Data

1. Results

III.1 Simulation Study Results

III.2 Application to Real Data Results

1. Conclusion

IV.1 Summary

IV.2 Limitations

IV.3 Future Improvements

REFERENCES

APPENDIX

**LIST OF TABLES**

TABLE

1. Adsf

**LIST OF FIGURES**

FIGURE

2.1 Something

**CHAPTER I**

**INTRODUCTION**

Introduction With growing advancements in technology that allow for more and more data to be recorded more intermittently. Growth of this time series data has led to increasing research in the field of functional data analysis, in particular development of regression models. However, due to the high dimensionality and the often highly non-linear nature of the underlying latent process, computational times can be excessive, especially when the functional data are non-Gaussian. Existing methods that model under this framework are sparse, with the exception of penalized flexiblefunctional regression (pffr) for exponential family data which is implemented in the refund package in the statistical software R via the refund::pffr() function Reference R package. However, this methodology does not scale well with increases in sample size and is not viable at larger values (N ¿= 500 in many cases). We propose a method, fast generalized functional on scalar regression (fGFoSR), that gives comparable or better estimation for fixed effects with small to moderate sample sizes (N ¡ 500) and accurate estimation and inference with substantially improved computation times for larg sample sizes. The basis of this method uses the concept from fast generalized functional principal component analysis (fGFPCA), [ref Andrew’s Paper?] and follows a form similar to that of a generalized linear mixed model:

All the coefficients vary over the function domain *s*, with being a fixed effect population mean, being a vector of fixed effect covariates, and being a subject specific random effect that represents an underlying latent process which is not directly observed, but estimated by leveraging the Karhunen-Loève expansion, . The expansion of is the sum of each underlying eigenfunctions and the mutually independent subject specific score ξik, over each of the *k* eigenfunctions. This results in the final model:

The advantage of this model is given not only in its comparable or better estimation and computation time to pffr, but the simplicity of implementation. Both of which will be shown using a (1) simulation study and (2) real world application.

**CHAPTER II  
METHODS**

**II..1 Fast Generalized Functional on Scalar Regression Implementation**

Fast generalized functional on scalar regression is simplistic in its implementation an can be broken down into the following 4 steps:  
Step 1. Binning of the data  
The first step in fGFoSR is to divide the data across the domain *S* into bins. To keep the model simple, we chose the width of each bin () to be equidistant and symmetric with the exception being on the boundaries of *S*. Using these prerequisites, the number bins (L) becomes the important decision in setup. A sample division is present below:

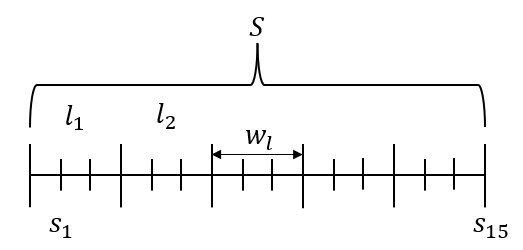


Figure 1. Binning example

An example of L = 5 equidistant bins (*l*) over the domain S which is subdivided into J = 15 subpoints (s). Each bin has a width () of . When choosing (L), it is important to consider the purpose of the bins in step 2, where a generalized linear mixed model is fit. With that in mind there needs to be few enough bins so that the domain is divided into large enough bin lengths, to capture sufficient data to fit an informed model and avoid convergence issues. Yet, the number of bins needs to be large enough to leverage the computational speed gain from multiple bins. The tradeoff between these two settings was not rigorously studied but more tested on a variety of ranges.

Step 2. Fit Generalized Linear Mixed Models (GLMMs)

Generalized Linear Mixed Models are then fit with each covariate and a random intercept  
in each bin l= 1,...,L to attain subject specific latent estimates. These models follow the form for ∈ . In this model is a fixed effect mean, is a vector of fixed effects based on the number of covariates in the data, and is a random intercept evaluated in the middle of the bin ( ). Using these models we get estimates of the global linear predictors for each subject. Since these estimates were derived from subsetted bins, they are not on the original domain, but rather the midpoints in , …, ⊂ *S*. This in turn makes the model misspecified since it assumes a constant effect over the points in each bin s j, when in fact , , and vary smoothly across it. This can cause biased estimations for , but under mild conditions. Furthermore, these biased estimates are only used to estimate the covariance matrix and eigenvectors in Step 3.

Step 3. Covariance Matrix/Eigenfunctions Estimation and Projection Onto the Original Do-  
main  
Using the estimated linear predictors from Step 2., we then estimate the covariance operator for using the fast covariance estimation (FACE) method from the refund package. Estimated eigenfunctions of are then obtained with the number of eigenfunctions K being a function of the percent variance explained, for our study purposes 95% was used. Lastly, because the eigenfunctions were estimated on the midpoints in , …, ⊂ *S* they need to be projected back onto the original domain. This is done using the reeval efunctions in the fastGFPCA package which estimates the eignefunctions at each point that the data was originally sampled.

Is the more I need to add here?

Step 4. Fit the Final Model With a Generalized Additive Model

With the projected estimated eigenfunctions, we arrive at our final model:  
Here (s) and (s) are smooth functions and the are mutually independent N∼(0,). Meaning that if we condition on the estimated eigenfunctions we have a generalized linear mixed model with K random slopes. Because the variance is a function of the K eigenvectors and the overall random slopes are uncorrelated, the computation of the model is relatively simple. Assuming a non-parametric model for our fixed effects, we can estimate the model using the mgcv::bam package using fast fREML smoothing and discrete = TRUE for computation efficiency. This process is intuitive and easily implemented:

Fig

**II.2 Simulation Study  
II.2.1 Simulation Design**  
For the simulation study, data was created using binary functions on a J length grid in the  
domain S = [0,1] with J being equally spaced and shared across subjects. Subject specific curves  
were generated from the equation:  
The fixed effects and were generated with randomly distributed error in combination  
with splines to simulate complex variation over the domain, S. Random effects were generated  
through the subject-specific scores. We assumed K=4 principal components and the g(·) is the logit link. True eigenvalues are , with k = 1,2,3,4. The true eigenvectors were designed with a periodic element to mimic real data settings by following the form . For each subject and time point, binary observations were sampled independently from a Bernoulli distribution with probability. The actual run of the simulation was performed by running an Rscript through the terminal on MAC with a Intel(R) Xeon(R) W-3223 CPU with a 3.50GHz processor. In addition, the computer was not dedicated solely to the running of the simulation study but was in continuous process for all batches of both fGFoSR and pffr.

**II.2.2 Comparison to Existing Approaches**

To benchmark the effectiveness of fGFoSR, we compared the modeling efficiency and accuracy of our simulated data, to that of penalized flexible functional regression (pffr) under varying conditions. Specifically, we ran tests with differing subjects (N = 200, 500, 1000), number of points in the domain (J = 200, 1000, 2000), and number of bins used in the fGFoSR process (L = 20, 30, 40) in all combinations [[ref results table with combinations]], the exception of pffr not using bins. To be thorough, 100 data sets (R) were simulated under each combination of N, J, and L. With 7 the magnitude of runs and increasing data size, models were discontinued if run times exceeded 18 hours. In the case of timing out future models with larger data sizes were not run. To keep comparison as neutral as possible no parallelization was implemented and both sets of models were run on the same computer under similar conditions. The two modeling techniques were then compared on computation time as a measure of efficiency and bias, mean squared error, and coverage as a measure of accuracy. Each of the efficiency measures were calculated for both fixed effects and . For each R these measures were calculated as follows:

Text

Description automatically generated

Figure 2. Equations for Bias, MSE, and Coverage

Bias was calculated as the difference between the true simulated functions and the  
predicted outcome at each point in the domain (*j*) for each subject (*n*) and then summed. The mean squared error was calculated in a similar fashion but squaring the difference before summing over every subject. Lastly, coverage was calculated by first summing the number of points on the domain (*j*) that were within 95% confidence of the true value, using the standard error of the model and a the standard deviation of a normal distribution () for each subject. That sum is then divided by the total number of points in the domain (J) giving the coverage for that individual. To get the coverage of the sample, that value is then averaged over all the subjects (N). This was done for each fixed effect as denoted by the subscript in.

**II.3 Application to Real Data**

To further showcase the validity and flexibitiy of the fGFoSR method we applied the process to a subset of the National Health and Nutrition Survey (NHANES)ref this source data. This subset contained 8763 subjects each with 1440 points in the domain S ∈ [1, 1440]. The total number of covariates was also subsetted to include only 2, one binary and one continuous. This additional continuous covariate deviates from the simulation study and shows the flexibility and scalability of fGFoSR to take on more predictors. We processed the data with a set 30 bins a decision based on the number of points in the domain and kept the percent variance explained for estimating eigenvectors at 95%.

**CHAPTER III**

**RESULTS**

**III.1 Simulation Study Results**

**III.1.2 Comparison fGFoSR to PFFR**

Full results of the simulation are presented in Table 1 below. They show a summary of the measurements of accuracy averaged across the 100 simulated data sets for both fGFoSR at varying bins. Notably, data for pffr does not exist at the highest level of data complexity for N = 200 and not all for N > 200. This is a result of the pffr method not being able to complete 100 runs within 18 hours that was as a reasonable time to completion. For the runs completed we see close to, if not better, average model bias, mean squared error, and coverage between the two models at all bin count variations for fGFoSR.

Table 1. Bias, Coverage, and MSE for 100 Runs, fGFoSR vs PFFR at N = 200

Graphical user interface, application

Description automatically generated

Furthermore, we can visually see consistent estimates of the true values by fGFoSR for increasing number of observation points (*J*) for both fixed effects and , see Figure 3 and Figure 4 respectively below as well as for increasing number of subjects (*N*), see Figure 5 and Figure 6. In line with Table 1 we see a closer estimation to the true values as J and N increase.

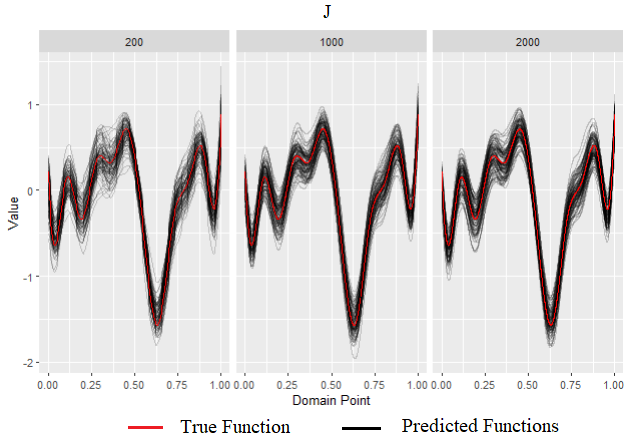


Figure 3. fGFoSR predicted vs true f(0) across increasing number of observations J, for N=200, L=30

Chart, histogram

Description automatically generated

Figure 4. fGFoSR predicted vs true f(1) across increasing number of observations J, for N=200, L=30

With the accuracy established to the baseline of PFFR, we can now turn our attention to the improvement upon processing time. Of the three varying simulation parameters (J, N and B), N has the greatest effect on processing time. Looking at computation times in Table 2 for N = 200 between the two models, fGFoSR outperforms pffr on a magnitude of 50x faster at J = 200 and 30x faster at J = 1000. Furthermore, fGFoSR is capable of processing datasets with even more subjects, N = 500, at comparable times to pffr at N = 200, Figure 5.

Table 2. Runtime in minutes by number of observations J.

Graphical user interface, application

Description automatically generated

Chart, bar chart

Description automatically generated

Figure 5. Runtime comparison between fGFoSR and pffr at varing N.

**III.1.2 fGFoSR Across Increasing Data Complexity**

Due to lack of results from pffr for higher numbers of subjects, results

Lastly, when considering variations in bin count between fGFoSR runs with the same J and N, we see a trade-off between run time and accuracy Figure scatter x = runtime, y = bias, group = bin count; one for each N. This result is logical in that considering more bins would take longer but would allow for more information to be accounted for in the model estimation process.

Chart, bar chart

Description automatically generated

Figure 6. fGFoSR MSE for Increasing N Across J

Furthermore, there is a noticeable increase in model accuracy with respect to bias and MSE as the number of bins increase for fGFoSR method. This effect is magnified as the number of observation points on the domain (J) increases. However, the coverage does not have the same noticeable change, yet hovers around an impressive 95%. Added table that shows coverage across settings x = N, y = Coverage, group = JB This an indication that fGFoSR performs equal to if not better as far as accuracy is concerned.

Chart, bar chart

Description automatically generated

Figure 7. fGFoSR Runtime (min.) for Increasing N Across J

Chart, scatter chart

Description automatically generated

Figure 8. fGFoSR Coverage for Increasing N Across J

III.2 NHANES Results  
The results from processing the NHANES were computed in x hours and are presented as the  
predicted function in graph NHANES predicted values graph. The run time is line with projections from the simulation study when considering the large number of subjects, 8763, the large number of observations, 1440, and the addition of a third fixed effect. With the processing time still with a feasible length, the computational improvement over current methods is further reinforced.

**Chapter 4  
CONCLUSION**

**IV.1 Summary**

Fast generalized functional on scalar regression represents a significant improvement over the current established method, penalized flexibile functional regression. Results from the simulation study showed equal to, if not better, accuracy as well significant improvements to computation times. With pffr not being to process increasing complex data in a feasible time. The processing of the NHANES data flexed the ability of fGFoSR to handle additional covariates as well as large numbers of subjects and observations. Overall, fGFoSR represents a large step forward in the processing of functional data.

**IV.2 Limitations**

With computation time being a focused metric in this paper, a major limitation was based on the processing speed of the computer used. This speed is variable on the number of other processes running on it as well as several other factors including processor temperature. Additionally, the overall processing time is subject to the underlying true process, which was controlled using a seed across all simulations for one-to-one comparison purposes. Lastly is the limitation of time, it would be more informative to run more intermediary simulations between values for N, J, and B, but the value gained did not exceed the additional run time for more simulations.

**IV.3 Future Improvements**

Improvements aside, there are still several areas of consideration that could improve upon  
the work in this paper. Notably, implementing parallel processing to further increase computation times would be a beneficial area of research. Furthermore, a methodological way to determine optimal bin size and count could greatly benefit this method and help solidify the underlying ideas.