Manuscript progress report

Ying Jin

2023-12-05

Contents

Method

Assumptions

- For each subject i in the population, a generalized outcome $Y_i(t)$ is generated along a variable t (for example, time), where $t \in (0, T)$.
- The outcome, at any specific t, follows an exponential family distribution characterized by a (latent) continuous function $\eta_i(t)$:

$$g[E(Y_i(t))] = \eta_i(t) = \beta_0(t) + b_i(t)$$

$$p(Y_i(t)) = h(Y_i(t))exp\{\eta_i(t)T[Y_i(t)] - A(\eta_i(t))\}$$

- The continuous latent function consists of a population-level fixed process and an individual-level random process

$$\eta_i(t) = \beta_0(t) + b_i(t)$$

Observed data

In practice we would observe the discrete realization of $\{Y_i(t), t\}$ along a dense grid. For simplicity, we assume the observation grid is regular (same across sample). When we have J observations points in $(0, T]$, then for the jth observation point, we denote the corresponding value of t as t_j, and the corresponding outcome at this point $Y_i(t_j)$.

fGFPCA Algorithm

Bin data:

Choose a proper bin width w considering model complexity and identifiability. For now let's say the bins are equal-length and non-overlapping.

- Bin index $s = 1...S$
- Index of bin midpoints m_s
- Value of t corresponding to bin midpoints t_{m_s}
- Bin endpoints: $(t_{m_s} - \frac{w}{2}, t_{m_s} + \frac{w}{2}]$

Local GLMMs

At the every bin, we fit a local intercept-only model:

$$g[E(Y_i(t_j))] = \eta_i(t_{m_s}) = \beta_0(t_{m_s}) + b_i(t_{m_s})$$

where $t_j \in (t_{m_s} - \frac{w}{2}, t_{m_s} + \frac{w}{2}]$.

Here we are basically saying that the value of latent function is constant within the same bin, which clearly is a misspecification of the true latent process.

From the model above. we will be able to estimate a $\hat{\eta}_i(t_{m_s})$ on the binned grid for every individual in the training sample.

FPCA

Here, we fit a FPCA model on the $\hat{\eta}_i(t_{m_s})$ obtained from step 2:

$$\hat{\eta}_i(t_{m_s}) = f_0(t_{m_s}) + \sum_{k=1}^{K} \xi_{ik}\phi_k(t_{m_s}) + \epsilon_i(t_{m_s})$$

where ξ_{ik} independently follows normal distribution $N(0, \lambda_k)$, and $\epsilon_i(t_{m_s})$ at each point follows $N(0, \sigma_2)$.

From this model, we will be able to obtain the following estimates which are shared across population:

- Population mean $\hat{f}_0(t_{m_s})$
- Basis functions $\hat{\mathbf{\Phi}} = \{\hat{\phi}_1(t_{m_s}), ..., \hat{\phi}_K(t_{m_s}))\}$
- Estimates of variance of scores $\hat{\lambda}_1 ... \hat{\lambda}_K$

Projection and Debias

The mean and basis functions are evaluated on the binned grid. To extend it to the original measurement grid data was collected on, we project the estimated eigenfunctions $\hat{\mathbf{\Phi}}$ back use spline basis. Now we have extend the $\hat{\phi}_k(t_{m_s})$ to the original grid $\hat{\phi}_k(t_j)$

Because of the misspecification of local GLMMs, the estimated eigenfunctions and eigenvalues are also biased by a constant multiplicative effect. Therefore, we use a GLMM to re-evaluate the mean function, eigenfunctions and eigenvalues.

Out-of-sample prediction

Now, let's assume we have a new subject u with J_u observations ($J_u < J$). Then the log-likelihood of this new subject would be:

$$l_u = \sum_{t_j < t_{J_u}} log(h(Y_u(t_j))) + \hat{\eta}_u(t_j)T(Y_u(t_j)) - log(A[\hat{\eta}_u(t_j)])$$

where $\hat{\eta}_u(t_j) = \hat{f}_0(t_j) + \sum_{k=1}^{K} \xi_{uk}\hat{\phi}(t_j)$.

With estimates for the population-level parameters from fGFPCA algorithms above, we can estimate ξ_{uk} by maximization of l_u. Direct maximization some times does not have closed form solution. Numeric maximization methods seem not very stable as well. So I have decided to used a Bayes approach (Laplace Approximation):

- Prior distribution: $\xi_{uk} \sim N(0, \hat{\lambda}_k)$
- Posterior distribution: the likelihood of $l_u = l(Y_u(t_j)|\xi_u)$

Laplace Approximation would get the posterior mode of ξ_{uk} through quadratic approximation.

Larger-scale simulation

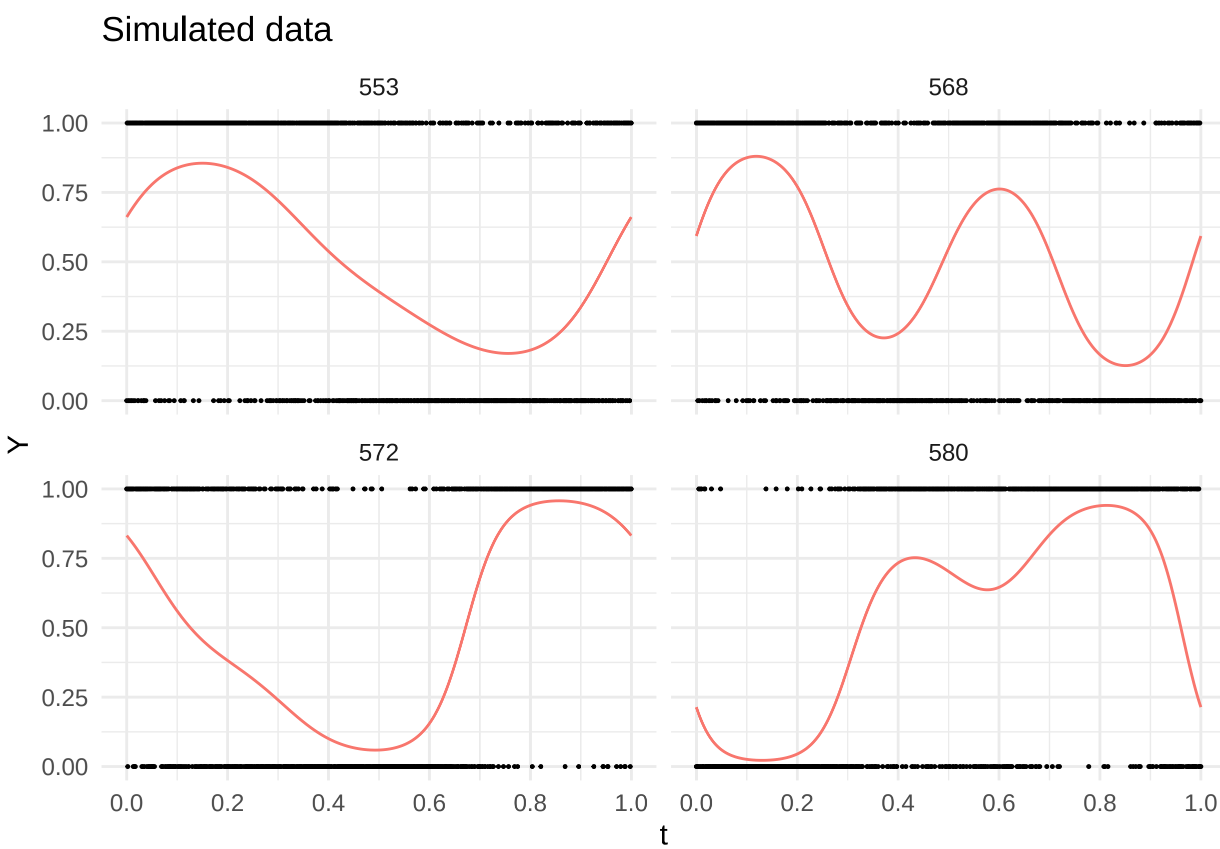
Simulation set up

Here we simulate binary data from cyclic latent process:

$$Y_i(t) \sim Bernoulli(\frac{exp(\eta_i(t))}{1+exp(\eta_i(t))})$$
$$\eta_i(t) = f_0(t) + \xi_{i1}\sqrt{2}sin(2\pi t) + \xi_{i2}\sqrt{2}cos(2\pi t) + \xi_{i3}\sqrt{2}sin(4\pi t) + \xi_{i4}\sqrt{2}cos(4\pi t)$$

where:

- t is 1000 equal-spaced observations points on $[0, 1]$ (J = 1000).
- $f_0(t) = 0$
- $\xi_k \sim N(0, \lambda_k)$, and $\lambda_k = 1, 0.5, 0.25, 0.125$ for k = 1, 2, 3, 4 respectively.
- Sample size $N = 500$
- In the binning step, we bin every 10 observations
- 500 simulations were implemented



Reference method

- GLMMadaptive
- Here we can fit a model with random intercept and slope for time. It is doable on 500 datasets, but obviously too simple for the data generation scheme. We would expect it to perform terribly.

$$g(E(Y_i(t))) = \beta_0 + \beta_1 t + b_{i0} + b_{i1} t$$

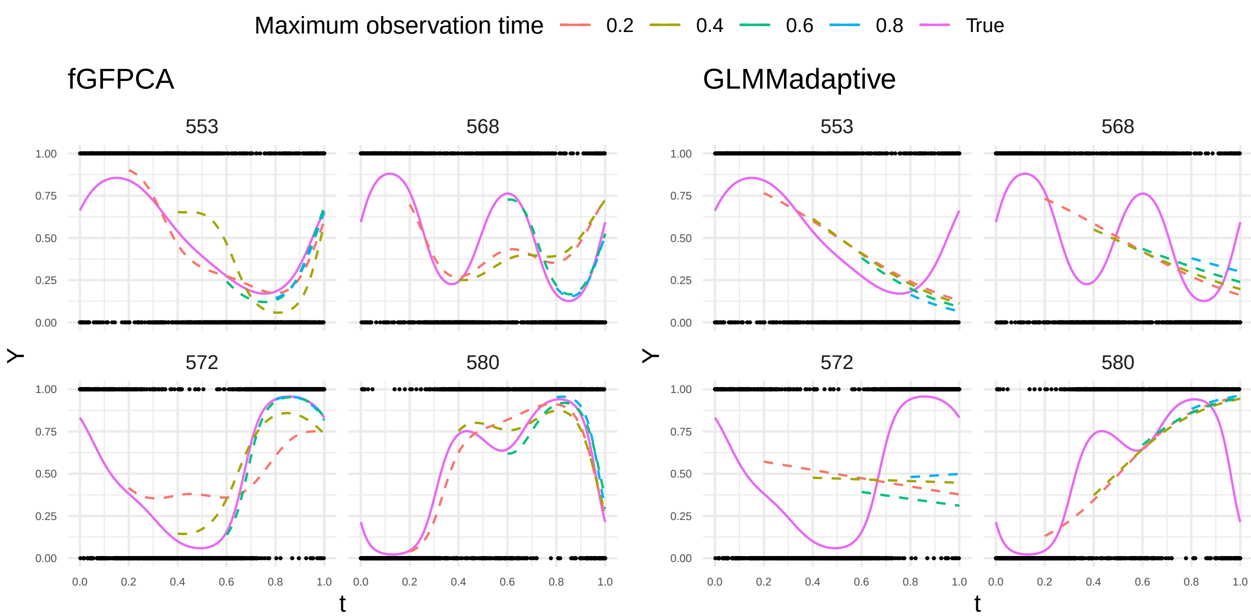
Table 1: Integrated squared error

	Maximum observation time							
	fGFPCA				GLMMadaptive			
Window	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8
(0.2, 0.4]	146.407				387.708			
(0.4, 0.6]	183.967	74.977			291.579	269.799		
(0.6, 0.8]	218.265	49.275	15.776		315.778	282.736	278.242	
(0.8, 1.0]	108.918	77.981	17.747	12.005	563.011	477.485	597.746	600.34

Table 2: Area under the ROC curve

	Maximum observation time							
	fGFPCA				GLMMadaptive			
Window	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8
(0.2, 0.4]	0.748				0.591			
(0.4, 0.6]	0.664	0.734			0.524	0.596		
(0.6, 0.8]	0.715	0.790	0.803		0.669	0.694	0.687	
(0.8, 1.0]	0.740	0.755	0.781	0.784	0.514	0.556	0.526	0.564

Figure



ISE

AUC

I think we could say that while the total time spend on model fitting + prediction are similar between two methods, fGFPCA achieved much better flexibility and much better predictive performance of prediction

Table 3: Computation time (minutes)

Method	Fit	Prediction
fGFPCA	0.725	1.592
GLMMadaptive	2.287	0.017

under every scenario.

Small-scale simulation: one iteration

Here we would like to fit fGFPCA and GLMMadaptive on a dataset with smaller sample size and/or smaller measurement density. For the GLMMadaptive model, we would set it up with spline basis functions so that its flexibility is comparable with fGFPCA model, such as:

$$g(E(Y_i(t))) = \sum_{k=1}^{4} \zeta_k B_k(t) + \sum_{l=1}^{4} \xi_{il}\phi_l(t)$$

I have used 100 subjects for training and testing. When fitting GLMMadpative, I reduce the number of measurements to 1/10 by taking one every 10 observations. The prediction is on the original grid.

Figure

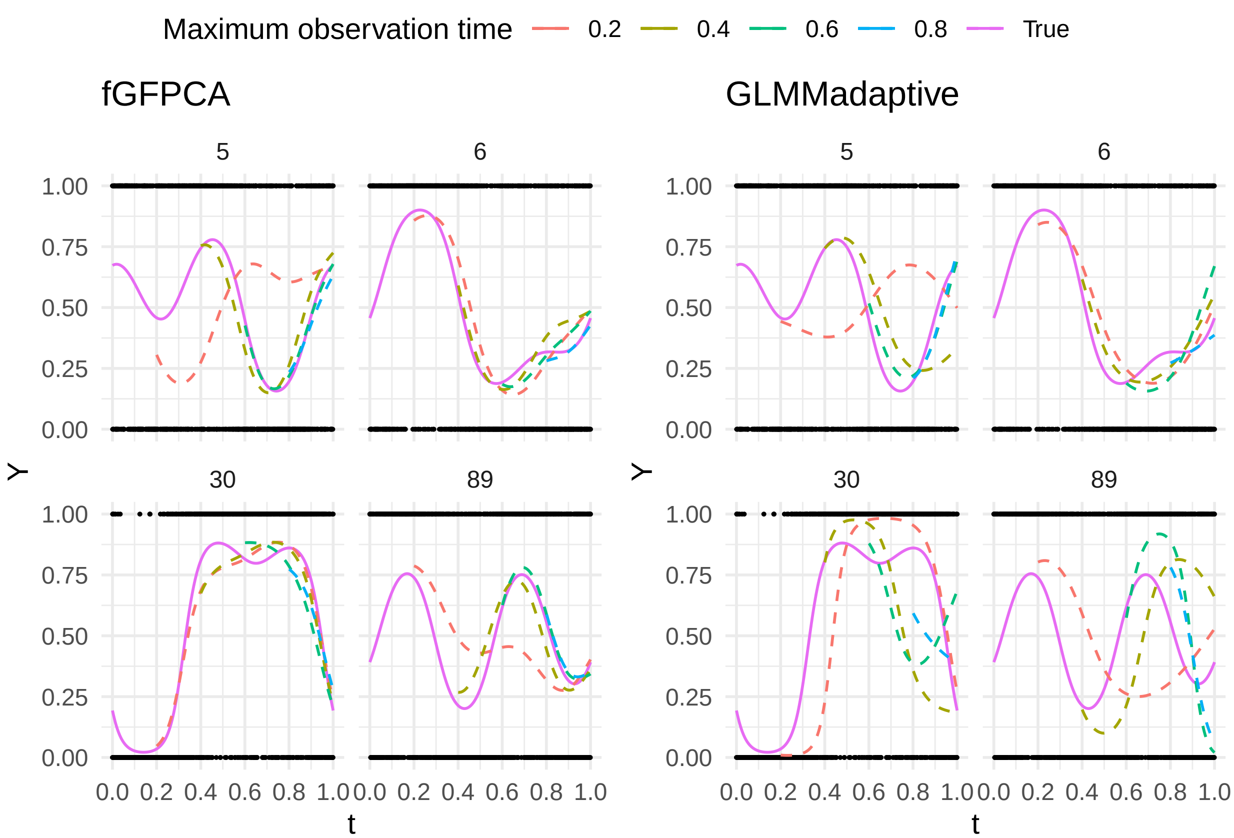


Table 4: Integrated squared error

	Observation track							
	fGFPCA				GLMMadaptive			
Window	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8
(0.2,0.4]	154.178				387.140			
(0.4,0.6]	168.881	70.248			248.395	415.355		
(0.6,0.8]	235.562	47.770	16.813		330.585	321.804	265.975	
(0.8, 1.0]	97.413	77.682	19.947	12.015	186.022	525.561	349.314	116.319

Table 5: Integrated squared error

	Observation track							
	fGFPCA				GLMMadaptive			
Window	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8
(0.2,0.4]	0.721				0.646			
(0.4,0.6]	0.658	0.725			0.620	0.665		
(0.6,0.8]	0.710	0.793	0.805		0.682	0.700	0.733	
(0.8, 1.0]	0.712	0.725	0.761	0.764	0.647	0.582	0.657	0.725

ISE

AUC

There is huge difference in compuation time. GLMMadaptive took 23 minutes, while fGFPCA took less than 3 seconds.

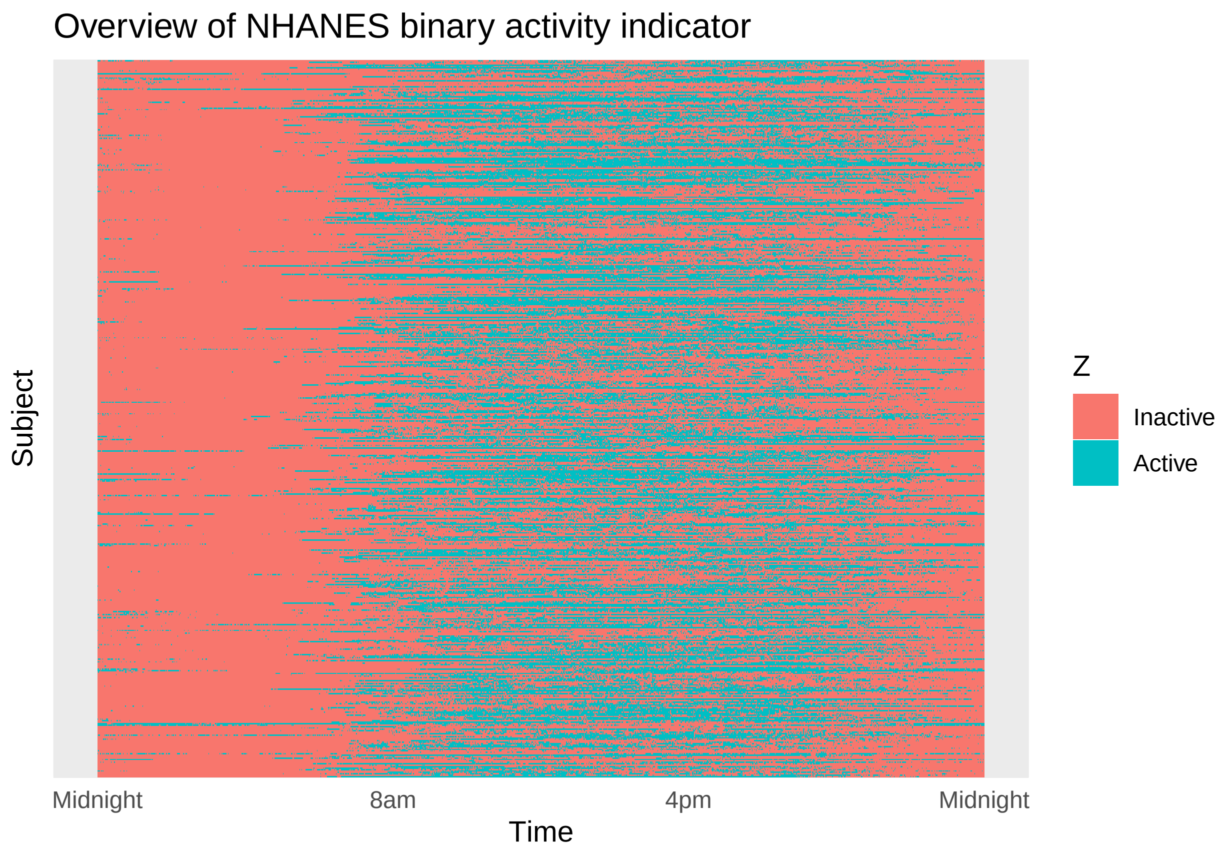
NHANES data application

We take 80% (7010) subjects for training, 20% (1753) for out-of-sample prediction.

In the GLMMadaptive model, I could only fit a model with random intercept. It took more than 20 minutes for model fitting.

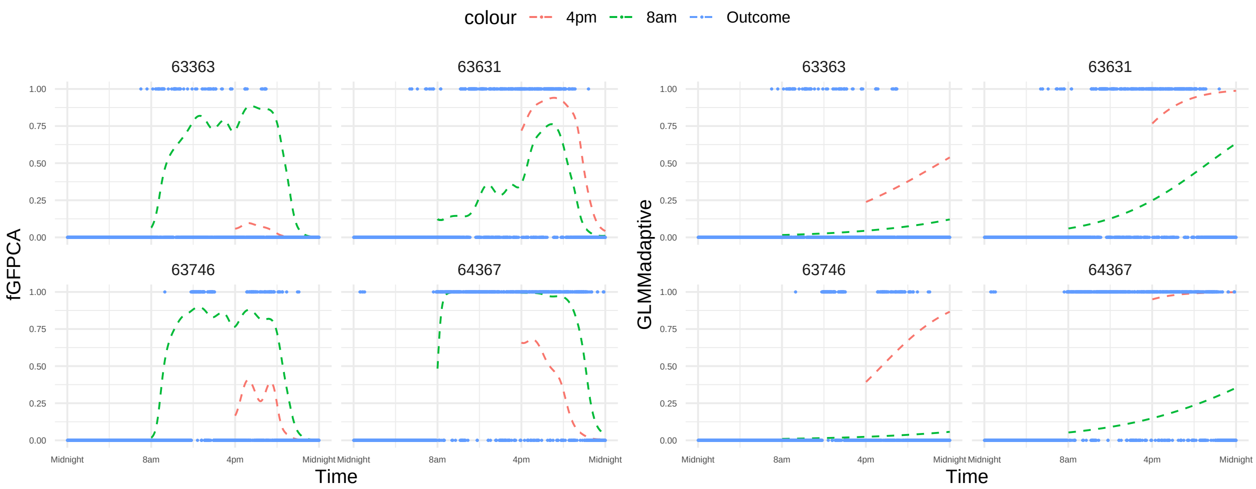
Table 6: Area Under the ROC curve

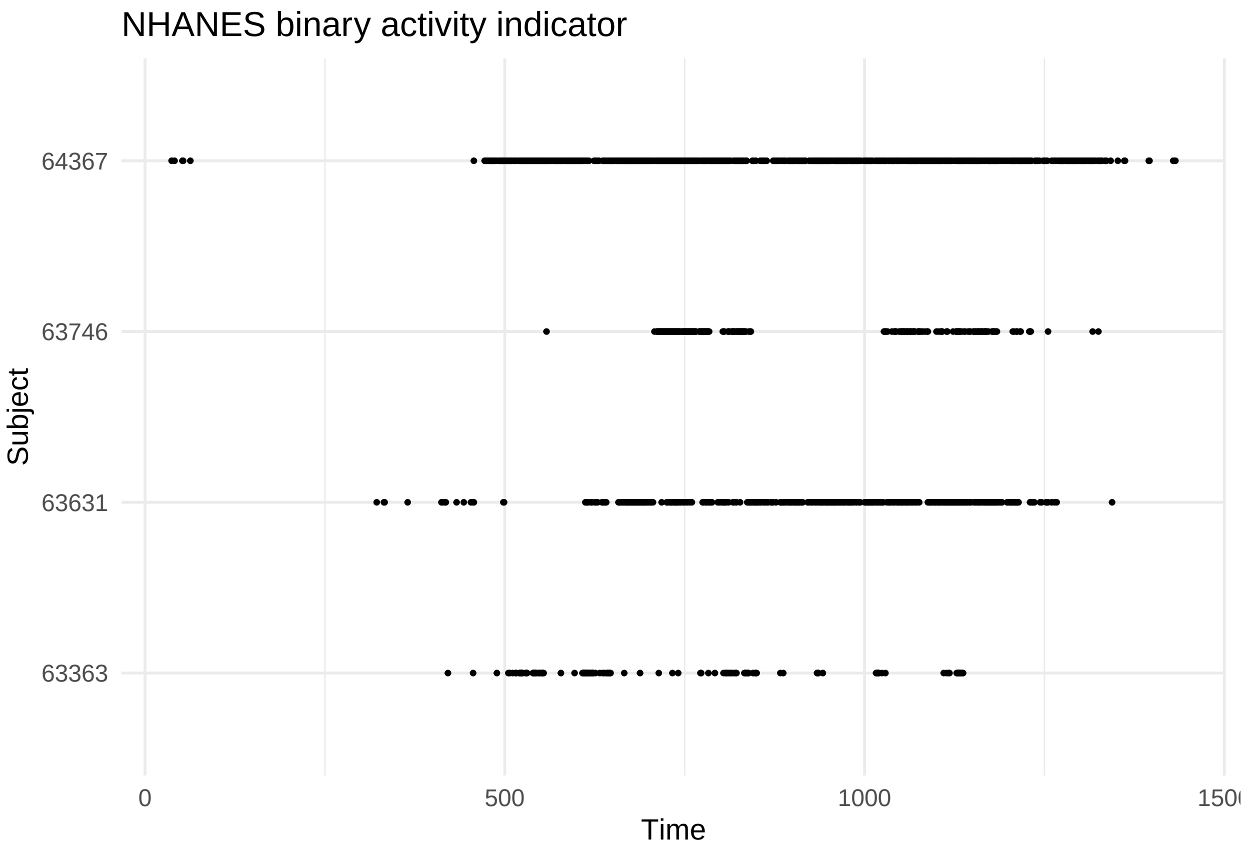
	Maximum observation time			
	fGFPCA		GLMMadaptive	
Window	8am	4pm	8am	4pm
8am-4pm	0.587		0.628	
4am-midnight	0.680	0.766	0.448	0.613



AUC

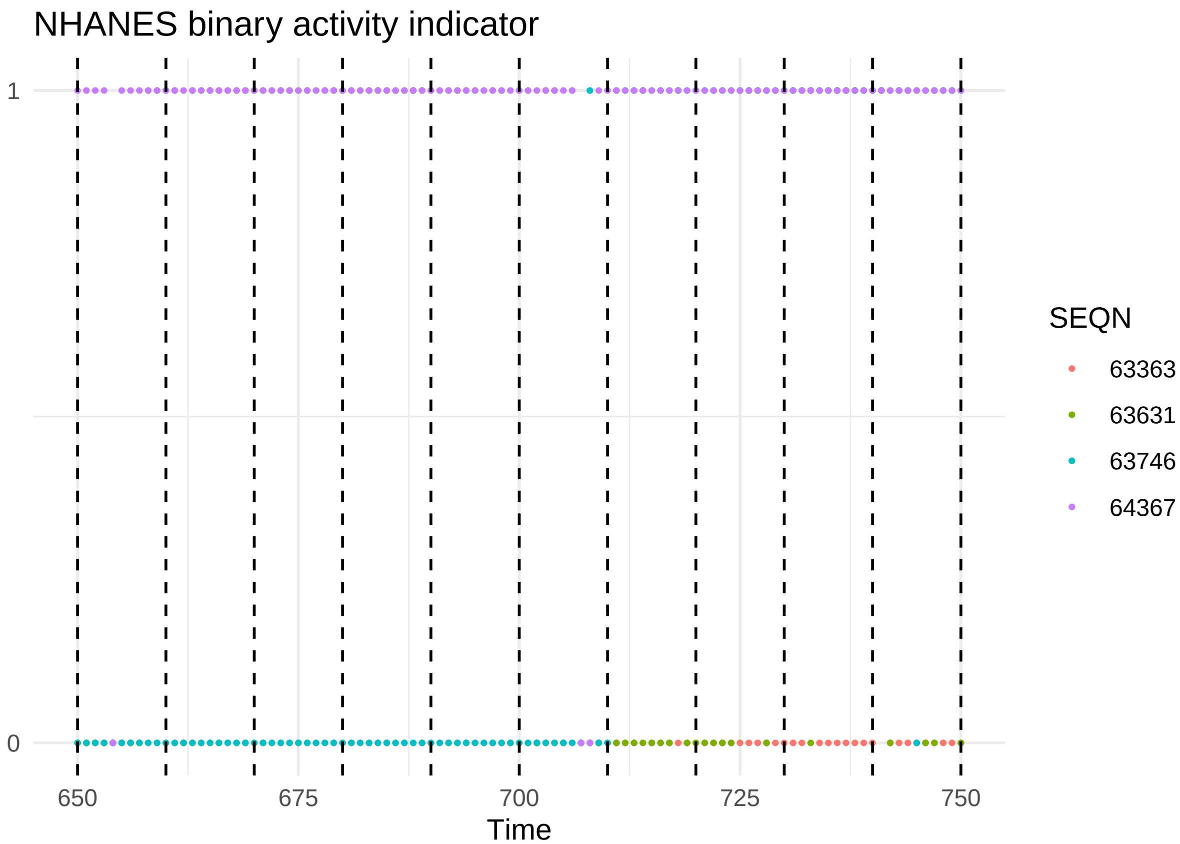
Figure

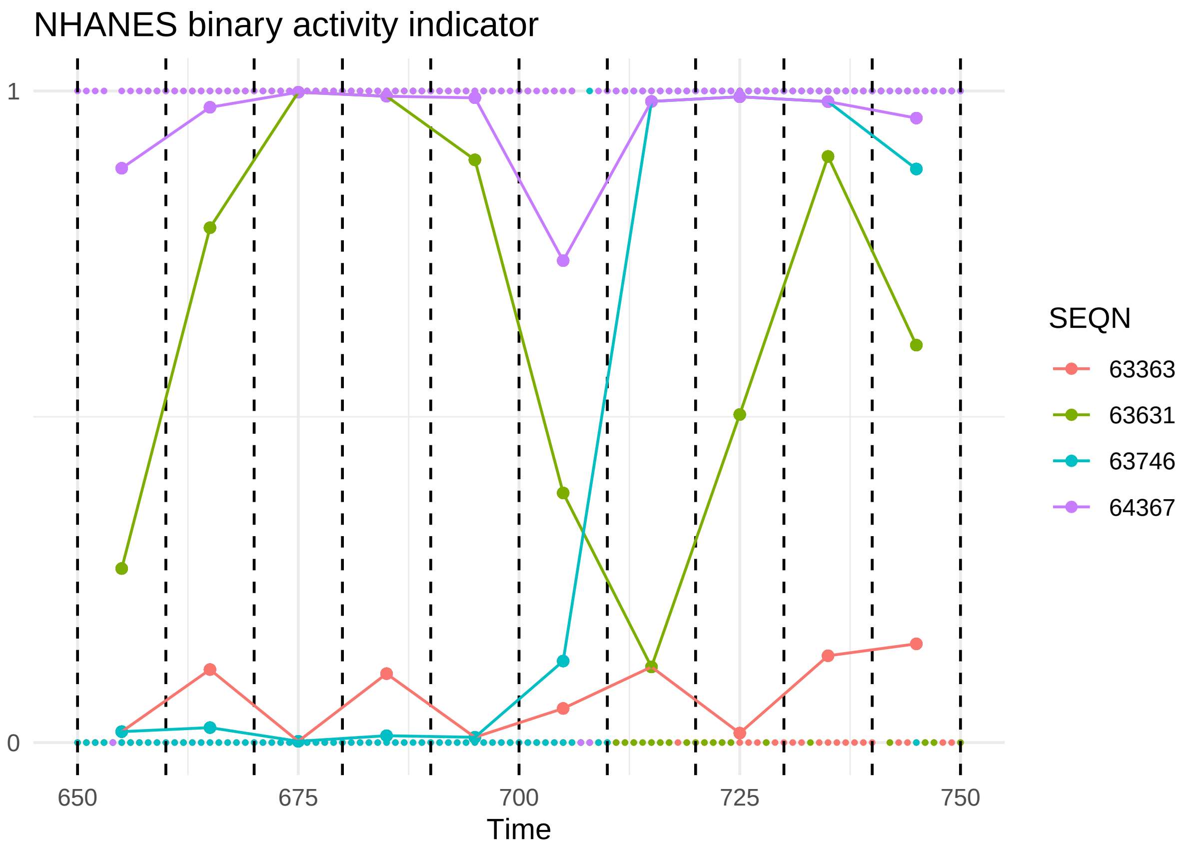

colour 4pm 8am Outcome
63363
63631
63746
64367
fGFPCA
GLMMadaptive
1.00
0.75
0.50
0.25
0.00
Midnight
8am
4pm
Midnight
Time


NHANES binary activity indicator
64367
63746
63631
63363
Subject
0
500
1000
150
Time

Additional figures

fGFPCA algorithm demonstration




NHANES binary activity indicator
1
0
650
675
700
725
750
Time
SEQN
63363
63631
63746
64367