Comparing different Sparse FDA functions in R

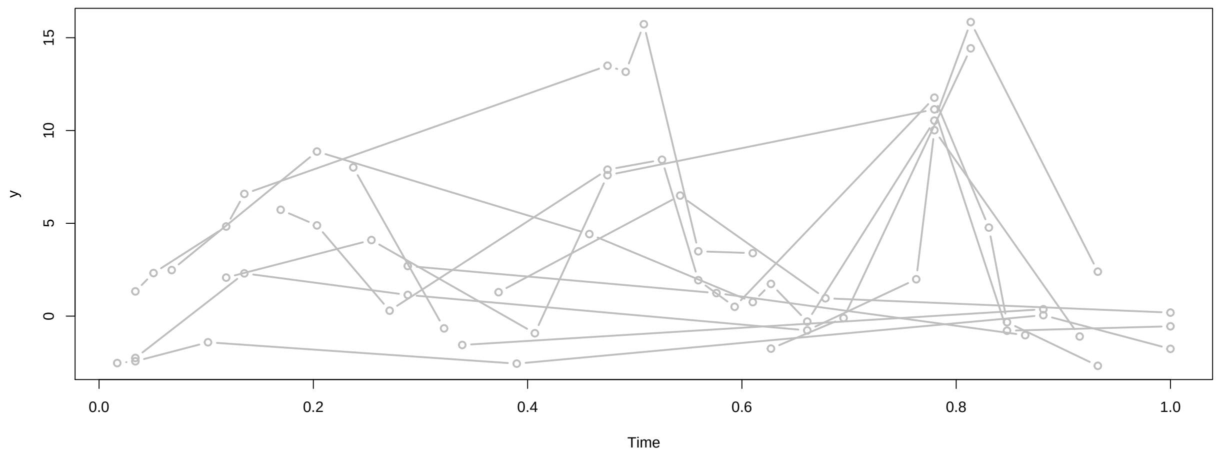
Tahmidul Islam

9/5/2020

Generate Sparse functional data.

```
rm(list=ls())
source("../code/functions_markdown.R")
#choose kernel
source("../code/RBF.R")
#source("code/laplace.R")
#source("code/matern52.R")
#source("code/matern32.R")



# data

# fdata_full <- fdagen(n = 20, gridSize = 60, sparsity = .1, muf = muf1)
# fdata1 <- fdata_full$sparseData
load('savedResusltComapre.Rdata')

testTime <- seq(0, 1, length.out = 100)
```

```
plotFdata(fdata1$id, fdata1$t, fdata1$y)
```



GP method

```
# Fit the GP model
#gpFit1 <- feature(id = fdata1$id, x = fdata1$t, y = fdata1$y)
postDist1 <- postDist(testTime, gpFit1)
```

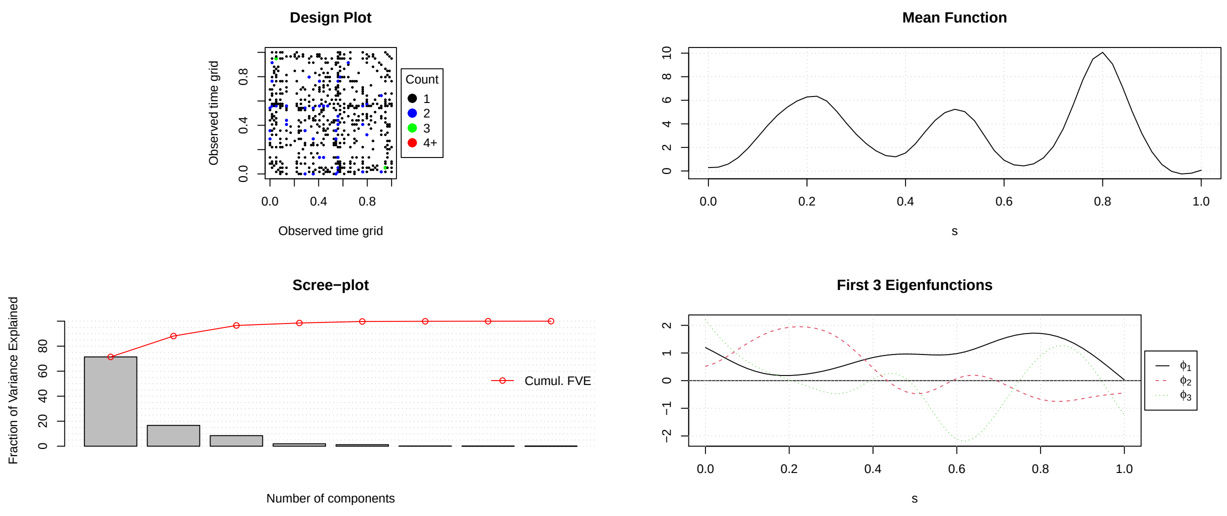
fdapace package : PACE and Local polynomial smoothing

Pacakge description: A versatile package that provides implementation of variousmethods of Functional Data Analysis (FDA) and Empirical Dynamics. The core of thispackage is Functional Principal Component Analysis (FPCA), a key technique forfunctional data analysis, for sparsely or densely sampled random trajectoriesand time courses, via the Principal Analysis by Conditional Estimation(PACE) algorithm. This core algorithm yields covariance and mean functions,eigenfunctions and principal component (scores), for both functional data andderivatives, for both dense (functional) and sparse (longitudinal) sampling designs.For sparse designs, it provides fitted continuous trajectories with confidence bands,even for subjects with very few longitudinal observations. PACE is a viable andflexible alternative to random effects modeling of longitudinal data.

Reference article: Yao, Fang, Müller, Hans-Georg and Wang, Jane-Ling (2005). "Functional data analysis for sparselongitudinal data." Journal of the American Statistical Association 100, no. 470 577-590.

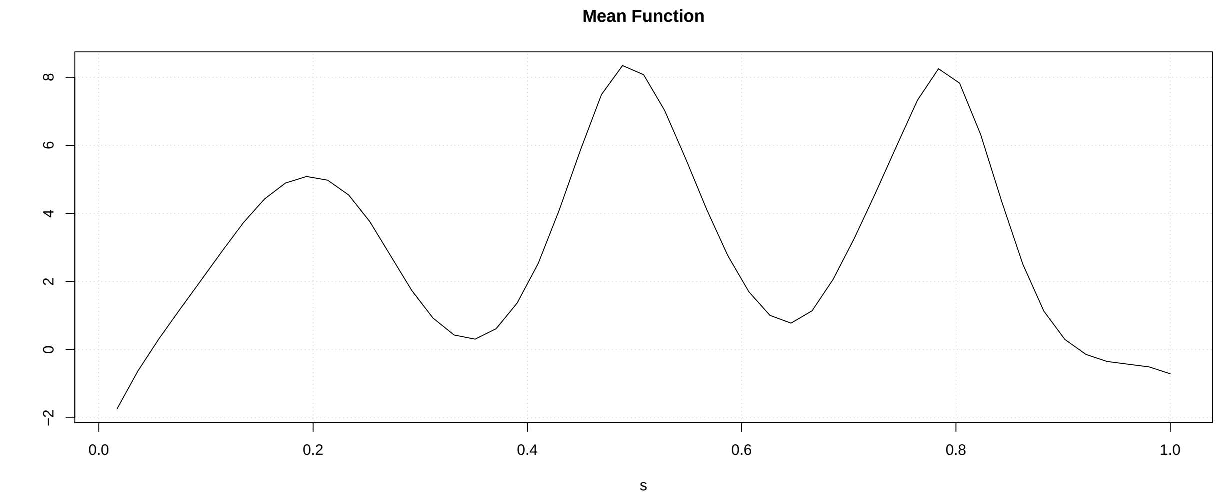
The overall summary plot from the package.

```
library(fdapace)
fpca.dt1 <- MakeFPCAInputs(IDs = fdata1$id, fdata1$t, fdata1$y, sort = FALSE)
fpca1 <- FPCA(fpca.dt1$Ly, fpca.dt1$Lt, optns = list(dataType = 'Sparse')) #fit model

plot(fpca1)
```

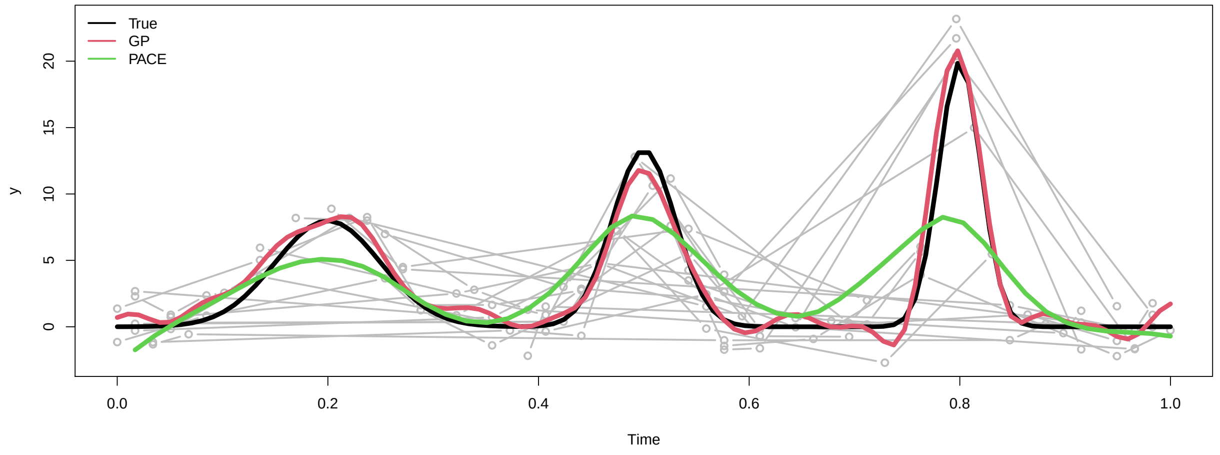


Mean function estimation from the PACE package

```
fm <- GetMeanCurve(Ly = fpca.dt1$Ly, Lt =fpca.dt1$Lt, optns = list(plot = TRUE))
```



```
fittedY <- fitted(fpca1, ciOptns = list(alpha=0.05))
plotFdata(fdata1$id, fdata1$t, fdata1$y)
lines(testTime, muf1(testTime), lwd = 5, col = 1)
lines(testTime, postDist1$mu, lwd = 5, col = 2)
lines(fm$workGrid, fm$mu, lwd = 5, col = 3)
legend('topleft', c('True','GP', 'PACE'), lty = 1, lwd = 2, bty = 'n', col = 1:3)
```

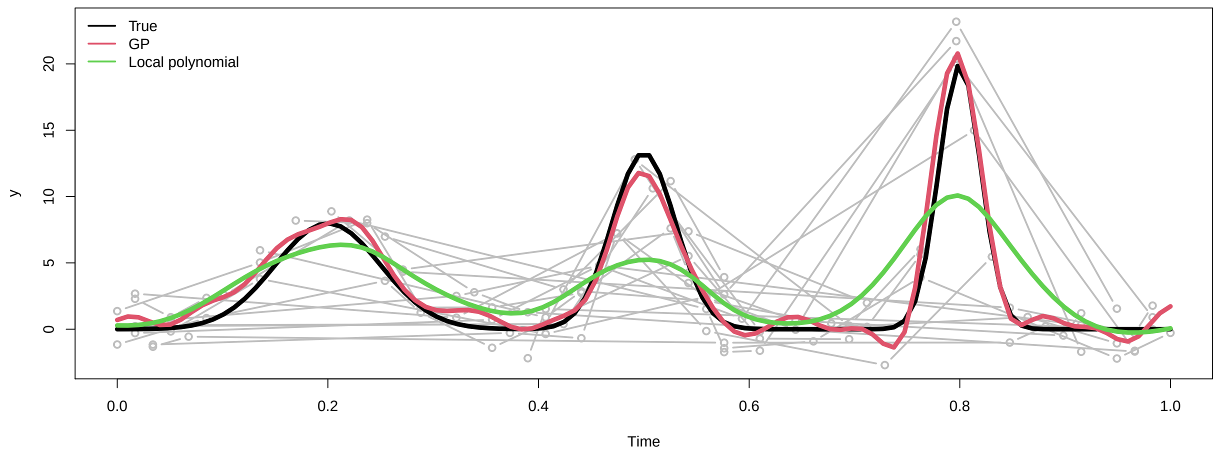


Local polynomial

PACE uses local polynomial smoothing for estimating the mean function. The local polynomial estimate is again obtained by manually aggregating the data and smoothing. This step does not involve FPCA.

```
sorted <- arrange(fdata1, t)
localp <- Lwls1D(bw = fpca1$bwMu, xin = sorted$t, xout = testTime, yin = sorted$y,kernel_type = 'gauss'

plotFdata(fdata1$id, fdata1$t, fdata1$y)
lines(testTime, muf1(testTime), lwd = 5, col = 1)
lines(testTime, postDist1$mu, lwd = 5, col = 2)
```

```r
lines(testTime, localp, lwd = 5, col = 3)
legend('topleft', c('True','GP', 'Local polynomial'), lty = 1, lwd = 2, bty = 'n', col = 1:3)
```



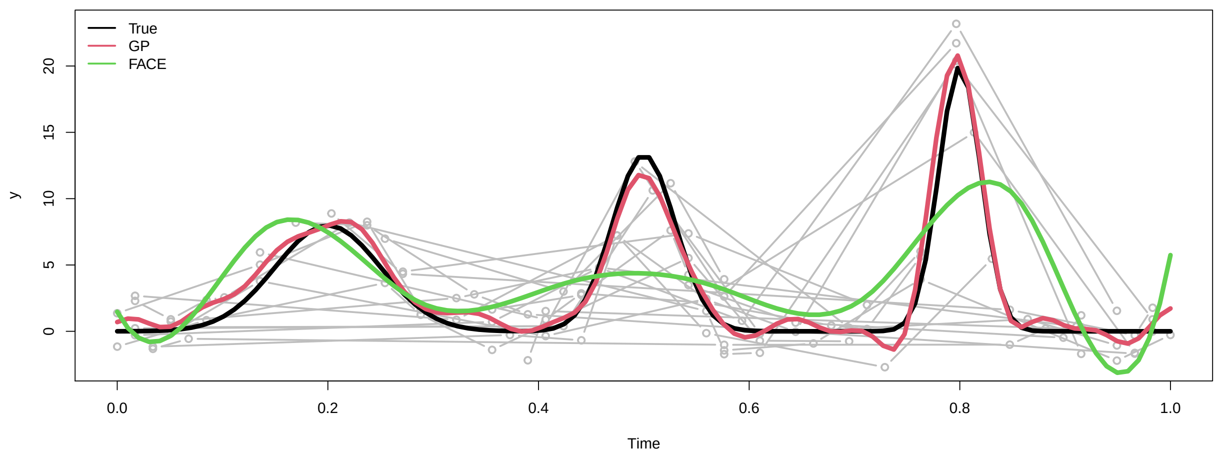
face package: P-spline

The face package is designed for fast covariance estimation for sparse functional data.

Package function description: This is a generalized version of bivariate P-splines (Eilers and Marx, 2003) for covariance smooth-ing of sparse functional or longitudinal data. It uses tensor product B-spline basis functions andemployes a differencing penalty on the assosciated parameter matrix. The only smoothing param-eter in the method is selected by leave-one-subject-out cross validation and is implemented with afast algorithm.There are two steps for estimation. During the first step, the objective function to minimize is thepenalized least squares on empirical estimates of covariance function. During the second step, thecovariance between the empirical estimates (depending on the estimates of covariance function) areaccounted and thus a generalized penalized least squares are minimized.If center is TRUE, then a population mean will be calculated and is smoothed by univariate P-spline smoothing:pspline(Eilers and Marx, 1996). This univariate smoothing uses leave-one-subject-out cross validation to select the smoothing parameter.

Reference: Luo Xiao, Cai Li, William Checkley and Ciprian Crainiceanu, Fast covariance estimation for sparsefunctional data, Stat. Comput., doi: 10.1007/s1122201797448.

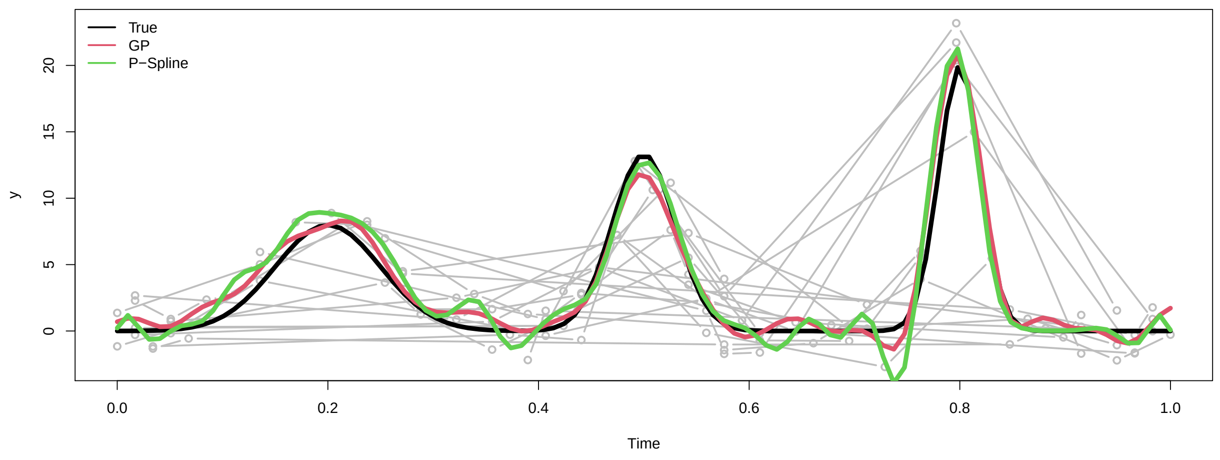
```r
library(face)

argvals <- fdata1$t
subj <- fdata1$id
y <- fdata1$y

data <- data.frame(argvals, subj, y)
fit_face <- face.sparse(data, argvals.new = testTime, center = T)

plotFdata(fdata1$id, fdata1$t, fdata1$y)
lines(testTime, muf1(testTime), lwd = 5, col = 1)
lines(testTime, postDist1$mu, lwd = 5, col = 2)
lines(testTime, fit_face$mu.new, lwd = 5, col = 3)
legend('topleft', c('True','GP', 'FACE'), lty = 1, lwd = 2, bty = 'n', col = 1:3)
```



P-Spline

Using package's own implementation of the P-spline, we estimate the mean function on the aggregated data.

```r
data <- data.frame(argvals, subj, y)
psp_pred <- pspline(data, argvals.new = testTime)

plotFdata(fdata1$id, fdata1$t, fdata1$y)
lines(testTime, muf1(testTime), lwd = 5, col = 1)
lines(testTime, postDist1$mu, lwd = 5, col = 2)
lines(testTime, psp_pred$mu.new, type = 'l', lwd = 5, col = 3)
legend('topleft', c('True','GP', 'P-Spline'), lty = 1, lwd = 2, bty = 'n', col = 1:3)
```

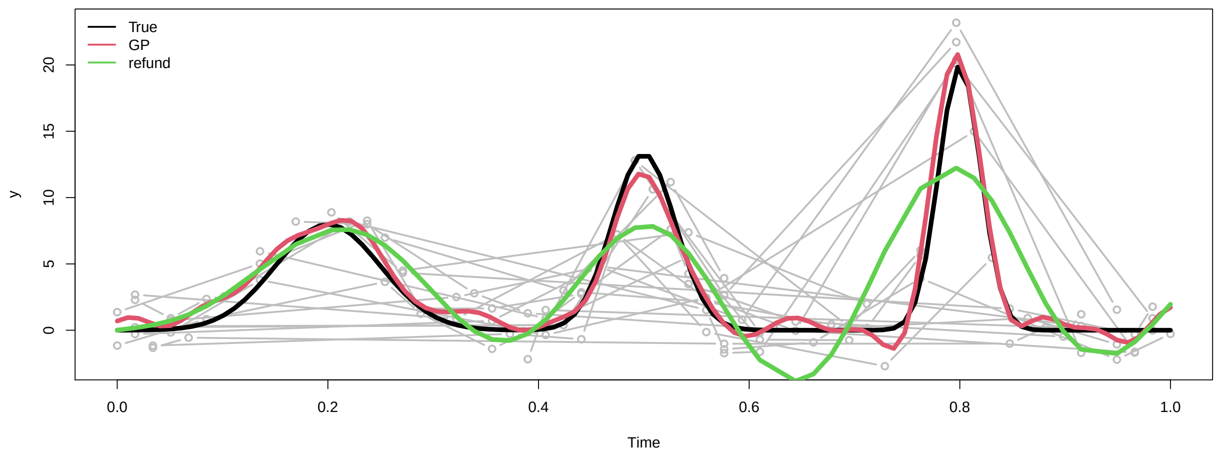


refund package: B-Spline

Package function description: This function computes a FPC decomposition for a set of observed curves, which may be sparsely observed and/or measured with error. A mixed model framework is used to estimate curve-specific scores and variances. FPCA via kernel smoothing of the covariance function, with the diagonal treated separately, was proposed in Staniswalis and Lee (1998) and much extended by Yao et al. (2005), who introduced the'PACE' method. fpca.sc uses penalized splines to smooth the covariance function, as developed by Di et al. (2009) and Goldsmith et al. (2013).

References: 01. Di, C., Crainiceanu, C., Caffo, B., and Punjabi, N. (2009). Multilevel functional principal compo-nent analysis.Annals of Applied Statistics, 3, 458–488. 02. Goldsmith, J., Greven, S., and Crainiceanu, C. (2013). Corrected confidence bands for functionaldata using principal components.Biometrics, 69(1), 41–51.

```r
library(refund)

ydata <- data.frame(.id = fdata1$id, .index = fdata1$t, .value = fdata1$y)
Fit.MM <- fpca.sc(ydata=ydata, var = TRUE)

plotFdata(fdata1$id, fdata1$t, fdata1$y)
lines(testTime, muf1(testTime), lwd = 5, col = 1)
lines(testTime, postDist1$mu, lwd = 5, col = 2)
lines(sort(unique(fdata1$t)), Fit.MM$mu, lwd = 5, col = 3)
legend('topleft', c('True','GP', 'refund'), lty = 1, lwd = 2, bty = 'n', col = 1:3)
```



No separate implementation of B-spline found in the package.

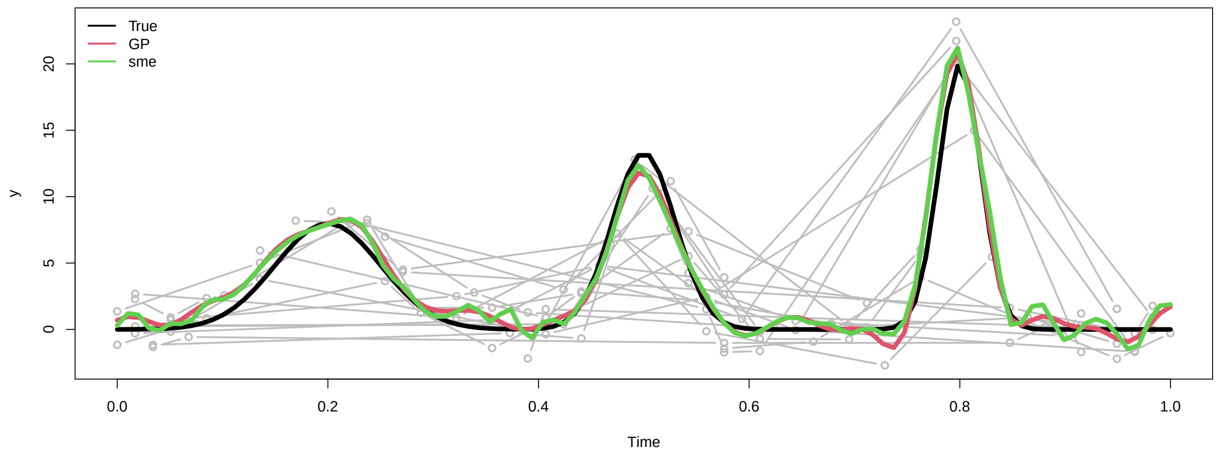
sme package: Smoothing spline mixed effect model

Package description: A package for fitting smoothing-splines mixed-effects models to replicated functional data sets.

Reference: Berk, M. (2012).Smoothing-splines Mixed-effects Models in R. Preprint

```r
library(sme)

fit_sme <- sme(data.frame(tme = fdata1$t, ind = as.factor(fdata1$id), y = fdata1$y),
               criteria = 'AIC', maxIter = 10000, initial.lambda.mu = 0, initial.lambda.v = 0)
sme_mu <- spline(x = as.numeric(colnames(fit_sme$coefficients)), y =  fit_sme$coefficients[1,], xout =
```

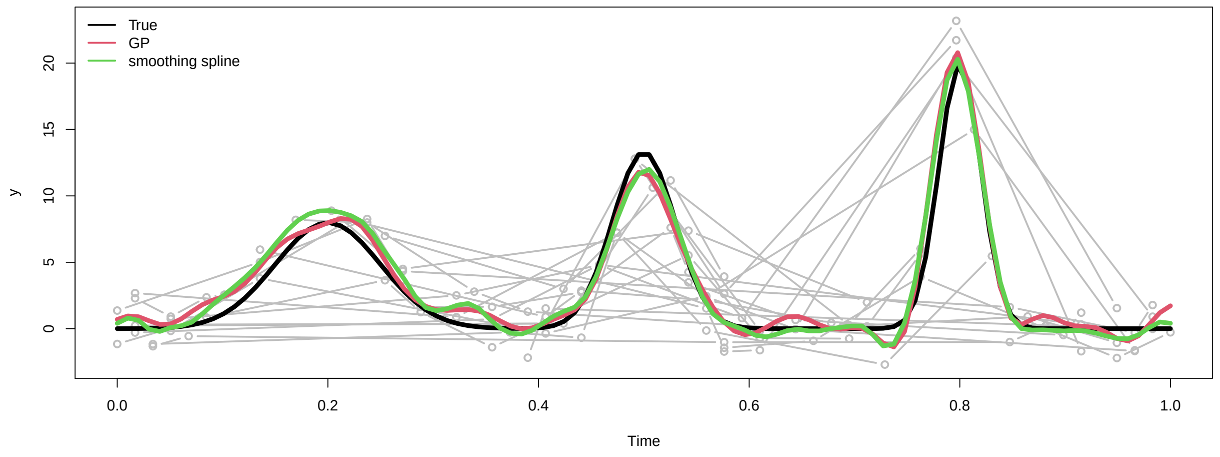
```r
plotFdata(fdata1$id, fdata1$t, fdata1$y)
lines(testTime, muf1(testTime), lwd = 5, col = 1)
lines(testTime, postDist1$mu, lwd = 5, col = 2)
lines(sme_mu$x, sme_mu$y, lwd = 5, col = 3)
legend('topleft', c('True','GP', 'sme'), lty = 1, lwd = 2, bty = 'n', col = 1:3)
```



Smoothing spline

The package uses R base smooth spline function. The mean function is now estimated using smoothing spline after combining data.

```
sm_sp <- smooth.spline(x = fdata1$t, y = fdata1$y)
sm_sp.mu <- predict(sm_sp, testTime)

plotFdata(fdata1$id, fdata1$t, fdata1$y)
lines(testTime, muf1(testTime), lwd = 5, col = 1)
lines(testTime, postDist1$mu, lwd = 5, col = 2)
lines(testTime, sm_sp.mu$y, lwd = 5, col = 3)
legend('topleft', c('True','GP', 'smoothing spline'), lty = 1, lwd = 2, bty = 'n', col = 1:3)
```


True
GP
smoothing spline
y
Time