BST223: Bike Sharing Data Analysis

Liela Meng[1]
[1]Department of Biostatistics, UC Davis

1401 Words Student ID:917843295

Keywords: FDA, Smoothing, FPCA, Functional concurrent regression

8th February, 2021

Abstract

In this analysis, data from the bike-sharing system was analyzed to visualize the total count of bicycles per hour in January 2011 with the local linear kernel smoother and to investigate the relationship between feeling temperature and the count via functional concurrent regression model. Besides, smoothed curves of the first derivative of the count and three principal components were also provided.

1 Introduction

The bike-sharing system is a service for individuals to borrow a bike from a "dock" and return it at another dock (some systems are dockless). The renting system opens to either registered membership or casual customers.

Though the bike-sharing system's central concept is to serve as a public good and several kinds of research discovered positive economic and health impact, much of this critics concern about the negative consequences of those proliferated systems. In particular, with numerous bike-share programs available, the danger of oversupply has caught great attention, and some companies have focused on distributing bicycles effectively.

This analysis intends to visualize the hourly count of total rental bikes in January 2011, to investigate the dominant modes of variation, and to fit a functional concurrent regression model to estimate the relationship between feeling temperature and the renting counts.

The hypothesis of interest is that whether rush hours have more bike demands and the feeling temperature is positively related to the total demands. Thus, this analysis investigates how feeling temperature is related to the total demand. If the results are reliable in general, this analysis can serve as a guidance for redistributing bicycles.

2 Background

The data file "hour.csv" is downloaded from the UCI Mashing Learning Repository website [1]. This dataset contains the hourly count of rental bikes between the years 2011 and 2012 in the Capital bike-share system with the corresponding weather (e.g., temperature, wind speed, humidity, etc.) and seasonal information (e.g., holiday, weekend, etc.).

For the purpose of this analysis, only 2011 January's hourly data were used, and the selected variables are hour index, day index, hourly feeling temperature, and hourly total renting counts.

3 Methods

3.1 Smoothing

Two smoothing methods were utilized and compared in this analysis: one-dimensional local linear kernel smoother and B-spline with penalty (see appendix A figure 5).

One-dimensional local linear kernel smoother

The kernel estimator can be derived by minimizing the localized squared error while fitting a local linear estimator within the bandwidth.

We can define

$$\hat{g}(x) = \hat{\beta}_0(x) : \text{argmin} \sum_{i=1}^{n} (y_i - (\beta_0 + \beta(X_i - x)))^2 K(\frac{x - X_i}{h}).$$

The Kernel function used in this analysis is the Expanechnikov function: $K(x) = \frac{3}{4}(1 - x)^2|_{[-1,1]}$.

Bandwidth h was subjectively selected based on the automatically selected bandwidth value, which smooths the mean function using the Generalized Cross-validation (GCV) method.

Function *Lwls1D* in R package *fdapace* was used in the analysis.

3.2 Functional principal component analysis (FPCA)

FPCA is a statistical method with the goal of dimension reduction by investigating the dominant modes of data variation.[2]

Define the covariance operator as $\sum(g) = \int_I \sum(s,t)g(s)ds$, for any function $g \in L^2$. Under mild assumption, the spectral decomposition of $\sum$ is $\sum(s,t) = \sum_{k=1}^{\infty} \lambda_k \phi_k(s)\phi_k(t)$ where λ_k are eigenvalues in descending order and ϕ_k are the corresponding orthogonal eigenfunctions.

Thus, we can approximate X_i with K terms: $X_{iK}(t) = \mu(t) + \sum_{k=1}^{K} A_{ik}\phi_k(t)$, where $A_{ik} = \int_I (X_i(t) - \mu_t)\phi_k(t)dt$ are the functional principal components (FPCs).

In this analysis, using Gaussian function as the kernel smooth basis function, setting bandwidth as used in section 3.1 (smoothing), function *FPCA* in R package *fdapace* provides mean function, scree plot and eigenfunctions with fraction-of-variance-explained (FVE) threshold 0.99.

3.3 Functional Concurrent Regression Model

Since only feeling temperature is interested in the analysis, functional concurrent regression model with a single covariate $X(t)$ is:

$$Y(t) = \beta_0(t) + \beta_1(t)X(t) + \epsilon(t), \ t = 0, 1 \ldots, 23. \qquad (1)$$

β_0 is non-random function (functional intercept) and β_1 is the non-random coefficient function (functional slope).

Model (1),varying-coefficients model, assumes the value of Y at time t are independent from $X(s), s \neq t$, aka $Y(t) \perp X(s)$, and $I_X = I_Y = \{0, 1 \ldots, 23\}$.

In the analysis, the smoothing kernel function is the Gaussian function. Bandwidth for smoothed mean function and smoothed covariance function were subjectively specified based on the automatically estimated smooth method.

Function *FCReg* in R package *fdapace* was used in the analysis.

4 Results

4.1 Smoothing Curves

One dimensional local linear kernel smoother

Since the automatically selected bandwidth value for smoothing the mean function using GCV is 1.15, the subjective bandwidth was chosen around this value. Eventually, bandwidth 2 was specified.

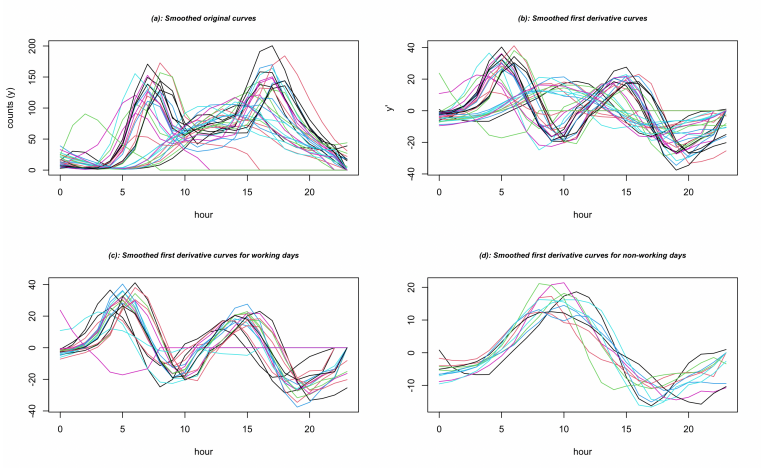

Fig. 1. One dimensional local linear kernel smoother

From figure 1 (a), where smoothing the original count of rented bikes by hour, we can see two peaks around 8 AM and 5 PM. This trend corresponds to people's typical commute time. Figure 1 (b) is the first derivative of the count, which seems to include two kinds of curves: the first one has peaks at hour 5 and 15, and the other one has one peak at 10 AM. It is natural to attribute the observed difference to whether that day is a working day or not. Hence, we can see from 1 (c, d), working days (c) have curves different from non-working days (d), and those two have the same characteristics as we discovered in figure (b).

4.2 FPCA

Auto-estimation method for mean and covariance functions, which was specified as *method-MuCovEst="smooth"* in function *FPCA*, gives bandwidth results for smoothing the mean and covariance functions as 1.15 and 2.3.

However, from figure 2, we can see the auto fitted plot (2(a)) fails to capture the surge at hour 8 and 17. Thus, I subjectively specified bandwidth for mean and covariance as 2 and 2.5, respectively (2(b)).

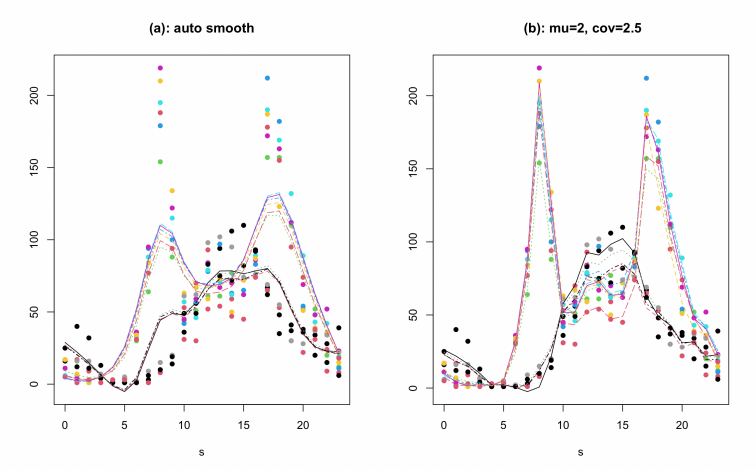

Fig. 2. Fitted sample path plot based on FPCA

Figure 3 shows the FPCA results. From the scree plot, we can see that three principal components (FPCs) explain more than 99% of the data's variance, and two FPCs explain more than 95%. The mean function still corresponds with what we discovered with the local linear kernel smoother (two peaks at hour 8 and 17). PCA1 shows a variation between rush hours and normal hours, while PCA2 shows variation between "brunch" time(11AM-4PM) and other times.

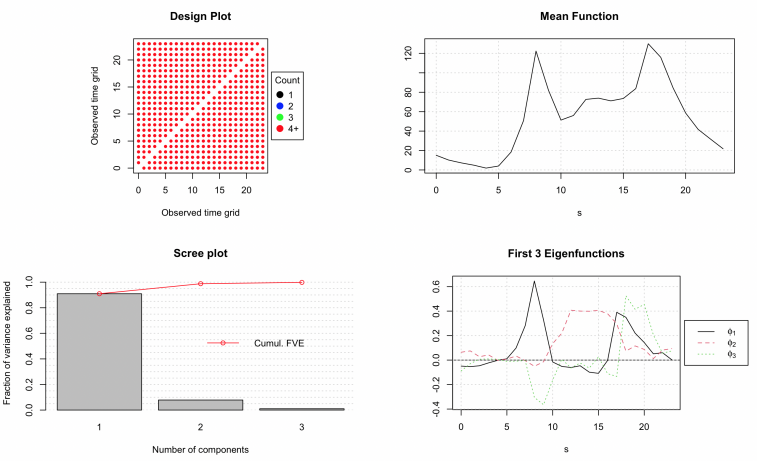

Fig. 3. FPCA results: design plot, mean function, scree plot, first 3 eigenfunctions

4.3 Functional concurrent regression

After setting bandwidth for mean and covariance as 2 and 2.5, respectively. Figure 4 shows the estimated intercept and slope per hour: expect for the time between hour 9 and 16, all three plots increase from hour 5 to 8 and decrease from hour 17 to 23. Overall, the feeling temperature is positively related to the total counts.

From hour 9 to 16, estimations of β_0, β (figure 4 (a,b)) show different pattern, which might indicate that in this time frame, the feeling temperature might play a more important role than in other times as the estimated coefficients are increasing at a relatively high level (see red dots in 4 (b)). However, coefficient estimations for working days and non-workings were also provided (see Appendix A figure 7,6) and two totally different patterns were observed. Such a split still failed to explain the exact relationship between temperature and the counts. Hence, we can not know for sure until we include more influential factors (e.g., wind speed, holiday index, etc.) into the model.

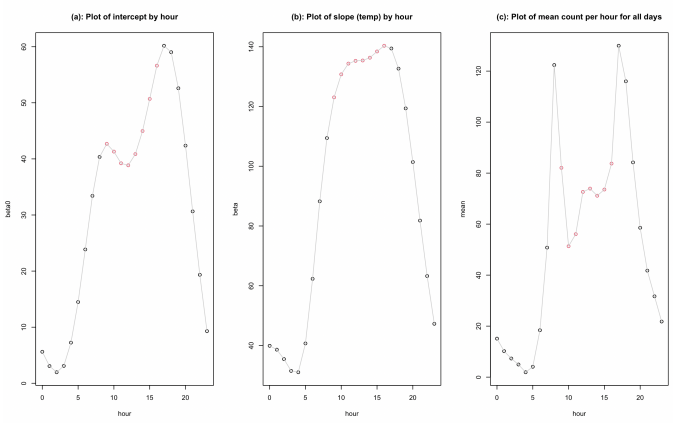

Fig. 4. Plot of estimated intercept and slope per hour

5 Discussion

After using the local linear kernel method, we can see that smoothed total count curves have two peaks around 8 AM and 5 PM (rush-hours) for working days and one peak around 10 AM for weekends, and those two features seem to correspond to PCA1 and PCA2.

The functional concurrent regression model was utilized to investigate the relationship between feeling temperature and the counts, and a positive relationship was observed across all times. In the time frame 9 AM to 4 PM, though we can see the estimated coefficients for temperature are increasing at a relatively high level, we can not explicitly explain how it is associated with the total count. Another thing to notice is that, the assumption $(Y(t) \perp X(s), s \neq t)$ of the functional concurrent regression model might not hold in reality.

Thus, I believe further research should focus more on including more influential factors and try the functional linear model rather than the varying-coefficient model.

6 Acknowledgement

I would like to thank Professor Hans-Georg Müller and Han Chen for their help in writing this report. I would like to extend my thanks to Professor Jiguo Cao for offering open course and R codes.

7 References

1. Fanaee-T, H. & Gama, J. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, 1–15 (2013).
2. Wang, J.-L., Chiou, J.-M. & Mueller, H.-G. *Review of Functional Data Analysis* 2015. arXiv: 1507.05135 [stat.ME].
3. Cao, J. *Functional Data Analysis Course* https://github.com/caojiguo/FDAcourse2019.

A Appendix

Penalized B-spline

The logic behind B-spline is: after selecting separating subintervals, piecewise polynomials defined on those subintervals are smoothly connected at these intervals' endpoints.

In order to control the smoothness of the fitted curve, a penalty function was added while minimizing the least-squares fits:

$$\tilde{Q}(\zeta_1, \ldots, \zeta_k) : \operatorname{argmin} \sum_{i=1}^{n} \{y_i - \sum_{k=1}^{K} \zeta_k B_k(t_i)\}^2 + \lambda P(\zeta).$$

In this analysis, roughness penalty was used as the penalty function $P(\zeta) = \int_\tau \{\sum_{k=1}^{K} \zeta_k B_k''(t)\}^2 dt$.

Function *eval.basis* in R package *dfa* was used and Composite Simpson's Rule was utilized to approximate the integral.

Setting smoothing parameter $\lambda = 0.8$, figure 5 (see appendix A) shows Penalized B-spline smoothing curves. Compared to figure 1 (a), though we can still observe two peaks around hour 9 and 17, the penalized B-spline smoothing smoother is less smooth than the local linear kernel smoother. Hence, the above analysis is based on the local linear kernel smoother. (Original R code used in this section see JiGuo Cao, Phd's github:[3])

Estimation in functional concurrent regression for working/non-working days

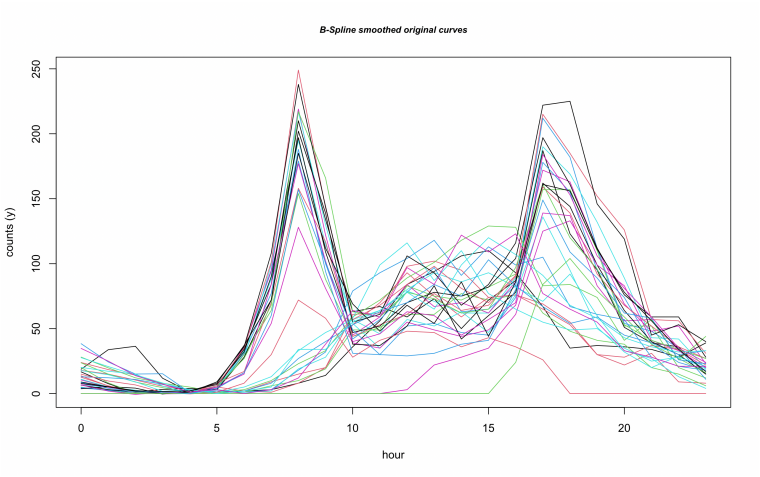

Fig. 5. Penalized B-spline smoothing smoother

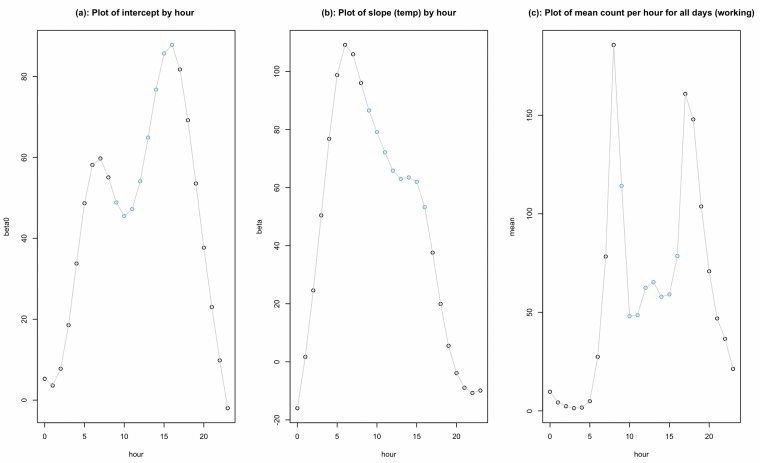

Fig. 6. Plot of estimated intercept and slope per hour for working days

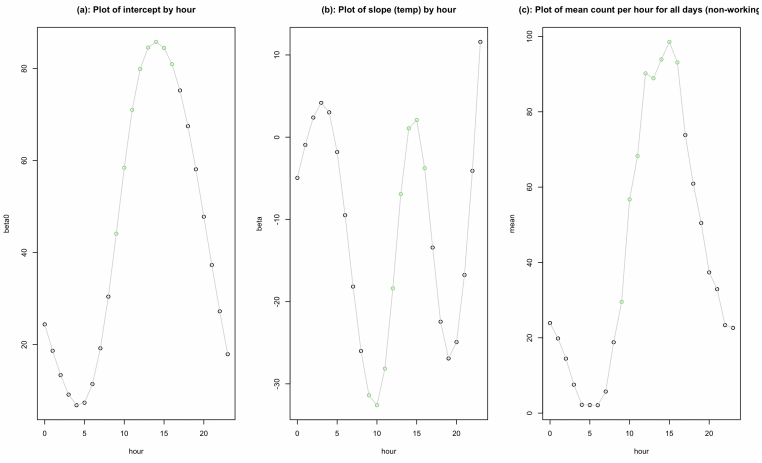

Fig. 7. Plot of estimated intercept and slope per hour for non-working days

B Appendix R code

```r
# Set up
   ------------------------------------------------------------------
library(readr)
library(fdapace)
library(tidyverse)
library(dplyr)
library(plyr)
library(fda)
library(lubridate)
library(ggplot2)
library(fdapace)
rm(list=ls())
hour <- read.csv("hour.csv")
data0 <- hour %>% filter(yr==0&mnth==1) %>%
  mutate(day=day(dteday)) %>%
  dplyr::select(hr,cnt,day)

data <- data0 %>% pivot_wider(
  names_from = day,
  values_from = cnt,
  values_fill = list(cnt = 0)
) %>% as.data.frame()
# Smoothing - 1D local linear kernel
   ------------------------------------------------------------------
## fit curves ##
# original curves
hour_df = hour %>% filter(yr == 0& mnth==1) %>%
  mutate(day=lubridate::day(dteday)) %>%
  dplyr::select(instant, cnt, day) %>%
  ddply(.(day), mutate, hours = 1:length(cnt),
        count_smooth = Lwls1D(bw = 2,
                              kernel_type = 'epan',
                              xin = hours,
                              yin = cnt,
                              xout = hours))
local_linear <- hour_df %>%
  dplyr::select(day,count_smooth,hours) %>%
  pivot_wider(
    names_from = day,
    values_from = count_smooth,
    values_fill = list(count_smooth = 0)
  ) %>% as.data.frame()%>% dplyr::select(-hours)

hour_df_der = hour %>% filter(yr == 0& mnth==1) %>%
  mutate(day=lubridate::day(dteday)) %>%
  dplyr::select(instant, cnt, day) %>%
  ddply(.(day), mutate, hours = 1:length(cnt),
        count_smooth=Lwls1D(bw = 4,
                            kernel_type = 'epan',
                            xin = hours,
                            yin = cnt,
                            xout = hours,
                            nder=1))

local_linear_der <- hour_df_der %>%
```

```
  dplyr::select(day,count_smooth,hours) %>%
  pivot_wider(
    names_from = day,
    values_from = count_smooth,
    values_fill = list(count_smooth = 0)
  ) %>% as.data.frame()%>% dplyr::select(-hours)
# workingday=1
hour_df_der_w1 = hour %>%
  filter(yr == 0& mnth==1 & workingday==1) %>%
  mutate(day=lubridate::day(dteday)) %>%
  dplyr::select(instant, cnt, day) %>%
  ddply(.(day), mutate,
        hours = 1:length(cnt),
        count_smooth=Lwls1D(bw = 4,
                            kernel_type = 'epan',
                            xin = hours,  yin = cnt,
                            xout = hours,
                            nder=1))

local_linear_der_w1 <- hour_df_der_w1 %>%
  dplyr::select(day,count_smooth,hours) %>%
  pivot_wider(
    names_from = day,
    values_from = count_smooth,
    values_fill = list(count_smooth = 0)
  ) %>% as.data.frame() %>% dplyr::select(-hours)
# working day=0
hour_df_der_w0 = hour %>%
  filter(yr == 0& mnth==1 & workingday==0) %>%
  mutate(day=lubridate::day(dteday)) %>%
  dplyr::select(instant, cnt, day) %>%
  ddply(.(day), mutate, hours = 1:length(cnt),
        count_smooth=Lwls1D(bw = 4,
                            kernel_type = 'epan',
                            xin = hours,
                            yin = cnt,
                            xout = hours,
                            nder=1))

local_linear_der_w0 <- hour_df_der_w0 %>%
  dplyr::select(day,count_smooth,hours) %>%
  pivot_wider(
    names_from = day,
    values_from = count_smooth,
    values_fill = list(count_smooth = 0)
  ) %>% as.data.frame()%>% dplyr::select(-hours)
## ploting ##
quartz()
par(mfrow=c(2,2))
# orginal
matplot(0:23,local_linear, type = "l", lty = 1, xlab='hour',
        ylab= 'counts (y)',main = list("(a): Smoothed original curves",
                                       cex = 0.8, font = 4))
# one derivative
matplot(0:23,local_linear_der, type = "l", lty = 1, xlab='hour',
        ylab='y\'',main = list("(b): Smoothed first derivative curves",
                               cex = 0.8, font = 4))
# working
```

```
matplot(0:23,local_linear_der_w1, type = "l", lty = 1, xlab='hour',
        ylab = '',main = list(
          "(c): Smoothed first derivative curves for working days",
          cex = 0.8, font = 4))
# non-working
matplot(0:23,local_linear_der_w0, type = "l", lty = 1, xlab='hour',
        ylab = '',main = list(
          "(d): Smoothed first derivative curves for non-working days",
          cex = 0.8, font = 4))
# Smoothing - Bspline
   ------------------------------------------------------
tobs = data[,1]
nobs = length(tobs)
knots    = c(seq(0,23,1));
nknots   = length(knots);
norder   = 4;
nbasis   = length(knots) + norder - 2;
basis    = create.bspline.basis(c(min(tobs),max(tobs)),
                                nbasis,norder,knots);
basismat = eval.basis(tobs, basis);
# Use quadrature to get integral - Composite Simpson's Rule
delta <- 0.02
quadpts <- seq(0,1,delta)
nquadpts <- length(quadpts)
quadwts <- as.vector(c(1,rep(c(4,2),(nquadpts-2)/2),4,1),mode="any")
quadwts <- c(1,rep(c(4,2),(nquadpts-1)/2))
quadwts[nquadpts] <- 1
quadwts <- quadwts*delta/3
# Second derivative of basis functions at quadrature points
Q2basismat = eval.basis(quadpts, basis,2);
# estimates for basis coefficients
Rmat = t(Q2basismat)%*%(Q2basismat*(quadwts%*%t(rep(1,nbasis))))
# estimates for basis coefficients
basismat2 = t(basismat)%*%basismat;
lambda = 0.8   # smoothing parameter
Bmat  = basismat2 + lambda*Rmat;
data1<-data %>% dplyr::select(-hr) %>% as.matrix()
chat = ginv(Bmat)%*%t(basismat)%*%data1;
yhat = basismat%*%chat;
yhat2 = basismat%*%ginv(t(basismat)%*%basismat)%*%t(basismat)%*%data1
quartz()
matplot(0:23, yhat, type = "l", lty = 1, xlab='hour', ylab= 'counts (y)
   ',
        main = list("B-Spline smoothed original curves", cex = 0.8,
   font = 4))
# FPCA
   ---------------------------------------------------------------------
a<- hour %>% filter(yr == 0& mnth==1) %>% mutate(day=lubridate::day(
   dteday)) %>% dplyr::select(hr, cnt, day)
abc <- MakeFPCAInputs(a$day, a$hr, a$cnt)
# mu=2, cov=2.5
fpca <- FPCA(abc$Ly,abc$Lt,list(plot = TRUE, userBwMu=2, userBwCov=2.5)
   )
# atuo - mu=1.15, ocv=2.3
fpca_auto <- FPCA(abc$Ly,abc$Lt,list(plot = TRUE, methodMuCovEst="
   smooth"))
fpca_auto$bwMu;fpca_auto$bwCov
#compare
```

```
quartz()
par(mfrow=c(1,2))
CreatePathPlot( fpca_auto, subset = 1:10, main = "(a): auto smooth",
    pch = 16)
CreatePathPlot( fpca, subset = 1:10, main = " (b): mu=2, cov=2.5", pch
    = 16)
# fpca
quartz()
plot(fpca)
CreateOutliersPlot(fpca, optns = list(K = 3, variant = 'KDE'))
CreateFuncBoxPlot(fpca, xlab = 'Hours', ylab = '# Renting', optns =
    list(variant='bagplot'))
SelectK(fpca,FVEthreshold = 0.99)
#fpca - derivative
der0 <- fitted(fpca,derOptns = list(p=1))
der <- t(der0)%>%  as.data.frame()
matplot(0:23,der, type = "l", lty = 1, main="FPCA: First derivation")
# FCReg
    ----------------------------------------------------------------
# data frame set up
data0 <- hour %>% filter(yr==0&mnth==1) %>% mutate(day=day(dteday)) %>%
  dplyr::select(hr,cnt,day)
data1 <- data0 %>% pivot_wider(
  names_from = day,
  values_from = cnt,
  values_fill = list(cnt = 0)
) %>% as.matrix()
data2 <-hour %>% filter(yr==0&mnth==1) %>% mutate(day=day(dteday)) %>%
  dplyr::select(hr,atemp,day)  %>% pivot_wider(
    names_from = day,
    values_from = atemp,
    values_fill = list(atemp = 0)
  ) %>% as.matrix()
# 31*24
counting <- t(data1[,2:32] )
tempture <- t(data2[,2:32])
t <- seq(0,23,by=1)
X_1 <- Sparsify(tempture,t,sparsity =24)
Y_1 <- Sparsify(counting,t,sparsity =24) #c(10:20)
vars_ <-list(X_1,Y=Y_1)
fcreg <- FCReg(vars_,2,2.5,t)
# visulization
intercept <- fcreg$beta0 %>% as.data.frame() %>%
  rename_at( vars(starts_with(".")),~ str_replace(., ".", "beta0"))%>%
  mutate(out=c(0:23))
slope <- t(fcreg$beta)%>% as.data.frame() %>%
  rename_at( vars(starts_with("V")),~ str_replace(., "V1", "beta"))%>%
  mutate(out=c(0:23))
visual <- intercept %>% inner_join(slope)
col=c(rep(1,9),rep(2,8),rep(1,7))
data_mean <- data0 %>% dplyr::group_by(hr) %>% dplyr::summarize(mean=
    mean(cnt))
attach(visual)
quartz()
par(mfrow=c(1,3))
plot(out,beta0,col=col, main="(a): Plot of intercept by hour", xlab = "
    hour");lines(out,beta0,col="grey80")
```

```
plot(out,beta,col=col, main="(b): Plot of slope (temp) by hour", xlab = "hour");lines(out,beta,col="grey80")
plot(data_mean$hr,col=col,data_mean$mean,main="(c): Plot of mean count per hour for all days (all)", xlab= "hour", ylab="mean");lines(data_mean$hr,data_mean$mean,col="grey80")
detach(visual)


# working==0
    ------------------------------------------------------------

# data frame set up
data0 <- hour %>% filter(yr==0&mnth==1&workingday==0) %>% mutate(day= day(dteday)) %>%  dplyr::select(hr,cnt,day)
data1 <- data0 %>% pivot_wider(
  names_from = day,
  values_from = cnt,
  values_fill = list(cnt = 0)
) %>% as.matrix()
data2 <-hour %>% filter(yr==0&mnth==1) %>% mutate(day=day(dteday)) %>% dplyr::select(hr,atemp,day)  %>% pivot_wider(
  names_from = day,
  values_from = atemp,
  values_fill = list(atemp = 0)
) %>% as.matrix()

# 31*24
counting <- t(data1[,2:11] )
tempture <- t(data2[,2:11])
t <- seq(0,23,by=1)
X_1 <- Sparsify(tempture,t,sparsity =24)
Y_1 <- Sparsify(counting,t,sparsity =24) #c(10:20)
vars_ <-list(X_1,Y=Y_1)
fcreg <- FCReg(vars_,2,2.5,t)
# visulization
intercept <- fcreg$beta0 %>% as.data.frame() %>% rename_at( vars(starts_with(".")),  ~ str_replace(., ".", "beta0")) %>% mutate(out=c(0:23))
slope <- t(fcreg$beta)%>% as.data.frame() %>% rename_at( vars(starts_with("V")),  ~ str_replace(., "V1", "beta"))  %>%mutate(out=c(0:23))
visual <- intercept %>% inner_join(slope)
col=c(rep(1,9),rep(3,8),rep(1,7))
data_mean <- data0 %>% dplyr::group_by(hr) %>% dplyr::summarize(mean= mean(cnt))
attach(visual)
quartz()
par(mfrow=c(1,3))
plot(out,beta0,col=col, main="(a): Plot of intercept by hour", xlab = "hour");lines(out,beta0,col="grey80")
plot(out,beta,col=col, main="(b): Plot of slope (temp) by hour", xlab = "hour");lines(out,beta,col="grey80")
plot(data_mean$hr,col=col,data_mean$mean,main="(c): Plot of mean count per hour for all days (non-working)", xlab= "hour", ylab="mean");lines(data_mean$hr,data_mean$mean,col="grey80")
detach(visual)

# working==1
    ------------------------------------------------------------
```

```r

# data frame set up
data0 <- hour %>% filter(yr==0&mnth==1&workingday==1) %>% mutate(day=
    day(dteday)) %>%  dplyr::select(hr,cnt,day)
data1 <- data0 %>% pivot_wider(
  names_from = day,
  values_from = cnt,
  values_fill = list(cnt = 0)
) %>% as.matrix()
data2 <-hour %>% filter(yr==0&mnth==1) %>% mutate(day=day(dteday)) %>%
     dplyr::select(hr,atemp,day)  %>% pivot_wider(
  names_from = day,
  values_from = atemp,
  values_fill = list(atemp = 0)
) %>% as.matrix()

# 31*24
counting <- t(data1[,2:20] )
tempture <- t(data2[,2:20])
t <- seq(0,23,by=1)
X_1 <- Sparsify(tempture,t,sparsity =24)
Y_1 <- Sparsify(counting,t,sparsity =24) #c(10:20)
vars_ <-list(X_1,Y=Y_1)
fcreg <- FCReg(vars_,2,2.5,t)
# visulization
intercept <- fcreg$beta0 %>% as.data.frame() %>% rename_at( vars(starts
    _with(".")),  ~ str_replace(., ".", "beta0")) %>% mutate(out=c(0:23)
    )
slope <- t(fcreg$beta)%>% as.data.frame() %>% rename_at( vars(starts_
    with("V")),  ~ str_replace(., "V1", "beta"))  %>%mutate(out=c(0:23))
visual <- intercept %>% inner_join(slope)
col=c(rep(1,9),rep(4,8),rep(1,7))
data_mean <- data0 %>% dplyr::group_by(hr) %>% dplyr::summarize(mean=
    mean(cnt))
attach(visual)
quartz()
par(mfrow=c(1,3))
plot(out,beta0,col=col, main="(a): Plot of intercept by hour ", xlab =
    "hour");lines(out,beta0,col="grey80")
plot(out,beta,col=col, main="(b): Plot of slope (temp) by hour", xlab =
     "hour");lines(out,beta,col="grey80")
plot(data_mean$hr,col=col,data_mean$mean,main="(c): Plot of mean count
    per hour for all days (working)", xlab= "hour", ylab="mean");lines(
    data_mean$hr,data_mean$mean,col="grey80")
detach(visual)
```