Predictions of Temperatures in Winter Months in Detroit based on Temperatures in a nearby city Chicago: An FDA Regression

Avery Wu

Advised by Professor Edward Rothman

DEPARTMENT OF STATISTICS

UNIVERSITY OF MICHIGAN – ANN ARBOR

Table of Contents

1, Abstract 2

2, Introduction 3

3, Materials 5

4, Method of Analysis 8

(1) B-spline Curve Fitting and Smoothing 8

(2) Linear Models for Functional Responses 11

(3) Landmark Registration 14

(4) Ordinary Least Squares Linear Model 17

5, Results 18

6, Acknowledgements 19

7, References 19

8, Appendix of Code 20

(1) Data Input 20

(2) B-spline Curve Fitting and Smoothing 23

(3) Linear Models for Functional Responses 26

(4) Landmark Registration 33

(5) Ordinary Least Squares Linear Model 47

1, Abstract

This study practiced the functional data analysis (F.D.A.) modeling method for estimating the winter daily mean temperature in Detroit using that from Chicago. We hypothesized that there exists a relationship in time scale between the temperatures of the two cities. To put it into test, we applied F.D.A. methods including B-spline curve fitting, as well as smoothing by sum of least squares and roughness penalty, to transform the observed data into functions. The method of constructing linear models for functional responses (functional regression) reveals a strong relationship between the subjects. Predictions values are given by applying time warping functions in the landmark registration approach. As a result, this study compared the accuracy between the predictions from the F.D.A. regression model and the ordinary least squares (OLS) linear model; we then analyzed the pros and cons regarding the two general methods, and gave suggestions in model choosing for further relevant studies.

2, Introduction

Exposure in extreme low temperature could make a huge impact both economically and socially. In the week of January 1st to 7th, 2014, there were 193 emergency department visits in Michigan with self-reported cold-related injury complaints, for an average of 27.6 visits per day (Michigan Government report, 2014). The situation became worse in 2015, in which the previous 114-year record for the coldest temperatures were shattered in a number of cities in Michigan, with one of them dropped to -39 degrees Celsius, only 5 degrees warmer than the North Pole (MLive Weather, 2015). The freezing temperatures lasted until May, and brought a 47 percent decrease in sweet cherry production.

The more accurate our weather forecasting is, the more prepared we will be when facing these extreme weather conditions. Over the years, meteorologists have been working diligently to develop better models for precise forecasting. Common approaches nowadays include the persistence method, which assumes that the conditions at the time of the forecast would be consistent, and the trends method, which determines the speed and direction of fronts' movement, the location of high and low pressure centers, cloud distribution, as well as precipitation.

There are also several other forecasting methods. The climatology approach takes the average of the weather statistics over many years to make predictions. The analog approach examines the forecasting scenario at present, and search for a similar situation in the past to take as an analog. There is also the numerical weather prediction approach, in which mathematical models that predicts on climatic

parameters such as temperature, pressure, air flow, and precipitation are adopted and calculated by super computers.

In this paper, we used a statistical method, functional data analysis regression models, to determine the quantitative relationship between the winter daily mean temperature in Detroit and that in Chicago. Functional data analysis models the discrete observed data into continuous differentiable functional curves, and provides several options to reveal the underlying functional relationship between seemingly irrelevant variables.

3, Materials

The data for this paper were part of the Federal Climate Complex Global Surface Summary of Day Data version 7, from the National Climatic Data Center – NNDC Climate Data Online. Here we selected daily recordings from Detroit (station number 720113, WBAN number 54829) and Chicago (station number 725300, WBAN number 94846).

The dataset contains daily climatic records from January 1st, 1973 to February 28th, 2015. Selected variables that are used in this paper includes the daily means of temperature, sea level pressure, station pressure, wind speed in knots, as well as the daily maximum wind gust reported in knots. All temperatures are in degrees Fahrenheit.

To study Detroit in the coldest time of the year, we used the meteorological definition of winter, which is the period that runs from December 1st to February 28th for a total of 90 days. These three months are the coldest times of the year in the northern hemisphere, and is different from the commonly used astronomical winter, which refers to the winter season and is based on when the sun reaches the most southern point on the globe.

Fig. 1 The wind direction distribution (from which the wind came from) in Detroit

As is shown in Figure 1, for the months of December, January and February, more than 80% of the wind came from the West direction, especially from around WSW (247.5° on compass) and SW (225° on compass). Given that the geographic

coordinates of Detroit is 42°19’53’’N, 83°02’45’’W, and Chicago is on 41°50’13’’N, 87°41’05’’W, we calculated the initial bearing from Detroit to Chicago is 263°23’12’’, between the directions of W and WSW. Hence, a large portion of wind in Detroit during the winter months comes from the direction of Chicago. Therefore, by intuition, there should be a relationship between the temperatures in these two cities. In this paper, we applied functional data analysis to model this relationship, and make predictions of the temperatures in winter months in Detroit using that in Chicago.

4, Method of Analysis

(1) B-spline Curve Fitting and Smoothing

We first used the B-spline method to functionalize the temperature data from the two cities in each winter. Using a set of functional building blocks ϕ_k, with $k = 1, \dots, \mathrm{K}$ as the basis functions, we defined a basis function expansion $x(t)$ as a linear combination of the building blocks:

$$x(t) = \sum_{k=1}^{K} c_k \cdot \phi_k(t) = \mathbf{c}' \cdot \phi(t),$$

in which $c_1, c_2, \dots, c_K$ are coefficients, and $\mathbf{c}$ the *K*-vector of them. Here $\mathbf{c}'$ is the transpose of $\mathbf{c}$.

Notice that by default, the B-spline function in R would smooth the data with the sum of localized unweighted least squares fitting method, in which it uses the following error model:

$$y_j = x(t_j) + \varepsilon_j = \mathbf{c}' \cdot \phi(t) + \varepsilon_j = \phi'(t_j) \cdot \mathbf{c} + \varepsilon_j,$$

with y_j as the jth true value in our observed data, and ε_j as the error term of our jth estimation. Using this error model, the unweighted least squares estimation, which is the sum of all error terms, is calculated as:

$$\mathrm{SMSSE}(\mathbf{y}|\mathbf{c}) = \sum_{j}^{n} \left[y_j - \sum_{k}^{K} \left[c_k \cdot \phi_k(t_j) \right] \right]^2 = \sum_{j}^{n} \left[y_j - \phi(t_j)' \cdot \mathbf{c} \right]^2.$$

In the matrix form, we have:

$$\mathrm{SMSSE}(\mathbf{y}|\mathbf{c}) = (\mathbf{y} - \mathbf{\Phi} \cdot \mathbf{c})'(\mathbf{y} - \mathbf{\Phi} \cdot \mathbf{c}) = ||\mathbf{y} - \mathbf{\Phi} \cdot \mathbf{c}||^2 \ ,$$

with the *n* by *K* matrix $\mathbf{\Phi}$ containing the values of the *K* basis functions $\phi(t)$ at the

n sampling points, and $\mathbf{y}$ is the vector of our observed discrete data that is to be smoothed.

Therefore, to minimize SMSSE, we took its derivative corresponding to the coefficient vector $\mathbf{c}$, and let the result be 0 to yield $2\mathbf{\Phi}\mathbf{\Phi}'\mathbf{c} - 2\mathbf{\Phi}'\mathbf{y} = 0$. By solving this equation, we calculated our estimation for $\mathbf{c}$ is $\hat{\mathbf{c}} = (\mathbf{\Phi}'\mathbf{\Phi})^{-1}\mathbf{\Phi}'\mathbf{y}$, and then we have the estimation for the response variable is $\hat{\mathbf{y}} = \mathbf{\Phi}\hat{\mathbf{c}} = \mathbf{\Phi} \cdot (\mathbf{\Phi}'\mathbf{\Phi})^{-1}\mathbf{\Phi}'\mathbf{y} = S_\phi \cdot \mathbf{y}$. The corresponding projection operator to the basis function set is $S_\phi = \mathbf{\Phi} \cdot (\mathbf{\Phi}'\mathbf{\Phi})^{-1}\mathbf{\Phi}'$. By choosing values in $\mathbf{c}$ to minimize SSE, the B-spline curve is smoothed.

For our model, since the winter temperature data in a certain location is not periodic, here we used B-splines instead of Fourier series to construct our basis. In order to do so, we defined a set of $K = 20$ B-spline basis functions over the subintervals of our observation to be polynomials of a fixed degree or order, with each day of the three-month winter period to be a knot.

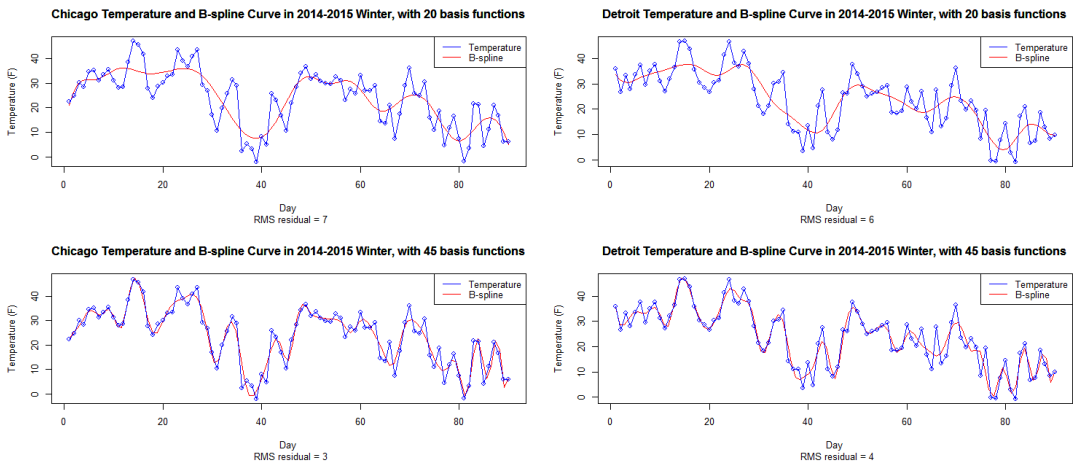

Fig. 2 The temperature line graph and the B-spline curves of Chicago (left) and Detroit (right) in the winter of 2014-2015, using 20 (upper) and 45 (lower) basis functions.

By smoothing the functions using our observed data, in this case is the daily temperature records from either Detroit or Chicago in a certain winter, we found the appropriate $\hat{\mathbf{c}}$ respectively and then received fitted B-spline curves, as is shown in Figure 2. The more basis functions we use, the more accurate is the resulting B-spline curve.

Fitting basis expansions by unweighted least squares means that we only have clumsy discontinuous control over the degree of smoothing. In order to get better results, we switched to a more powerful option, the roughness penalty smoothing approach, when a higher level of precision is needed later in this paper.

After functionalization of the data, we would like to check if there is a relationship between the daily temperatures in Chicago and Detroit in the winter months. We adopted 20 basis functions to capture the changing trend of the temperatures, and further we applied the B-spline curves to functional regression.

(2) Linear Models for Functional Responses

Using the B-spline curves, we applied the functional regression method to identify if there is a relationship of the winter temperatures between Detroit and Chicago. Since the direction of wind is from Chicago to Detroit, we treated the temperature in Chicago as the independent variable, and the temperature in Detroit as the response variable.

For each city in each year, we have modeled the temperature records as a B-spline basis function expansion $x(t)$. Therefore, our independent and response variables are both functional. Considering the distance between Chicago and Detroit, we would like to see a time lag between their temperatures. Hence, we have the ith linear model for our functional response is:

$$y_i(t) = \beta_0(t) + \int_{\Omega_t} x_i(s) \cdot \beta_1(t,s)\mathrm{d}s + \varepsilon_i(t),$$

in which the bivariate regression coefficient function $\beta_1(t,s)$ defines the dependence of $y_i(t)$ on the covariate function $x_i(s)$ at time t, and Ω_t is the range of values of s, in which $x_i(s)$ is considered influential to the value of $y_i(t)$. Notice that it is not necessary for $x_i(s)$ to be defined over the same range of time as $y_i(t)$ since the model is not concurrent.

For temperature prediction, let $x(s)$ to be the temperature of Chicago at time s, and $\mathrm{y}(t)$ to be the temperature of Detroit at time t. Set $\Omega_t = \{s|1 \leq s < t\}$ to imply forward causation. Therefore, by default of the R function, we used the integrated residual sum of squares as the unweighted fitting criterion, in which we integrated the sum of squares of $\varepsilon_i(t)$ over all models and all time. The formula is

shown as below:

$$\text{LMSSE}(\beta_0, \beta_1) = \int \sum_{i=1}^{N} \left[y_i(t) - \beta_0(t) - \int x_i(s) \cdot \beta_1(t,s) \ \text{d}s \right]^2 \text{d}t.$$

When the fitting criterion is minimized, the corresponding $\beta_0(t)$ and $\beta_1(t,s)$ are the functions that we desired. In the R environment, first we constructed three functional parameters for β_0, $\beta_1(t)$ and $\beta_1(s)$, and then we applied them as well as the B-spline basis function expansions for the temperatures of the two cities to the function `linmod` to get the finalized unweighted fitted model.

Notice that for better accuracy, we recalculated the B-spline basis function expansions using the roughness penalty smoothing approach. The smoothing parameter λ is chosen by the generalized cross-validation measure GCV, when the following criterion is at its minimum:

$$\text{GCV}(\lambda) = \frac{n^{-1} \cdot \text{SSE}}{\left[n^{-1} \cdot \text{trace}(\text{I} - \text{S}_{\phi,\lambda})\right]^2} = \left(\frac{n}{n - \text{d}f(\lambda)}\right)\left(\frac{\text{SSE}}{n - \text{d}f(\lambda)}\right)$$

in which the projection operator $\text{S}_{\phi,\lambda} = \boldsymbol{\Phi} \cdot (\boldsymbol{\Phi}'\boldsymbol{\Phi} + \lambda \mathbf{R})^{-1}\boldsymbol{\Phi}'$ is an order n symmetric matrix, and $\text{d}f(\lambda) = \text{trace}(\text{S}_{\phi,\lambda})$ is the degrees of freedom value for the spline smooth. The n by K matrix $\boldsymbol{\Phi}$ is the basis function values at a certain time t, as is defined previously in the B-spline Curve and Smoothing section.

Comparing to the sum of squares smoothing method we practiced in the B-spline Curve and Smoothing section, the only difference in the projection operator formulas is the addition of the penalty term $\lambda\mathbf{R}$ inside the parentheses of inverse. The penalty term would not be influential when $\lambda = 0$.

For our temperature data, the smoothing parameter λ is 1.65 for Detroit, and is

0.38 for Chicago. Using these values and the temperature data, we re-smoothed the B-spline basis function expansions, and with the beta functional parameters we constructed linear models for functional responses.

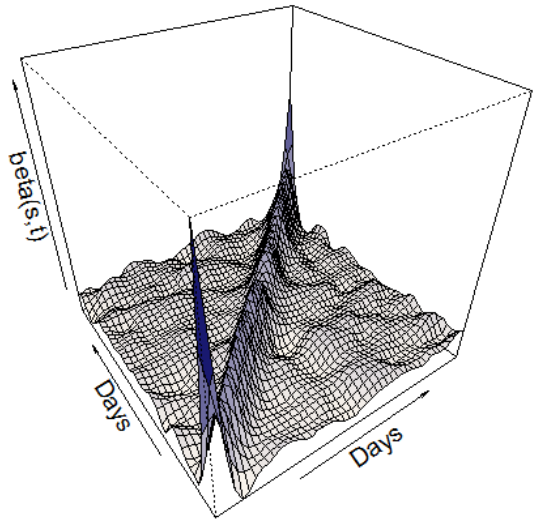

Fig. 3 The functional parameter function $\beta_1(t,s)$ for the prediction of Detroit 2014-2015 winter daily temperature from that in Chicago, estimated directly from observed climate data.

Figure 3 shows the estimated functional regression plane of $\beta_1(t,s)$ for the winter of 2014-2015, with the range of the estimated intercept function $\beta_0(t)$ about 4 orders of magnitude smaller than the response functions $y_i(t)$. Considering the magnitude of our original observed data, we took $\beta_0(t)$ to be essentially 0. The significant ridge at around the diagonal indicated that the winter daily temperature in Detroit is highly related to that in Chicago, with a time lag very close to 0. To investigate on the amount of time lag with even higher accuracy, we adopted the landmark registration method to find the time warping function of the two curves in a specific winter.

(3) Landmark Registration

In landmark registration, our particular focus is on the transformation of the time variable t, instead of on the values of $x(t)$ as we did in functional regression. Landmark registration emphasizes on two types of variability in the functional curves: amplitude and phase. Amplitude variability is the change of $x(t)$ in the vertical direction and not considering the value of t, phase variability is the change of t in the horizontal direction and not considering the value of $x(t)$.

For our temperature model, we identified the local extrema points as the landmark points. Therefore, the first order derivative of the B-spline curve at these points should all be equal to 0. In our case, the points were targeted by plotting the first order derivative function of the B-spline curve, and then applied the R function `locator` to hand-pick the locations of crossings of zero. A small error may be induced by this hand-selection method.

Mark the B-spline curve for Detroit temperature in a certain winter as $x_1(t)$, and that for Chicago as $x_0(t)$. For each curve $x_i(t),\ i = 0, 1$, mark the F landmark points that we identified as $t_{if},\ f = 1, \ldots, F$. We would like to find such a time warping function $h(t)$ that best satisfies:

$$x_2(t_{0f}) = x_1[h(t_{1f})].$$

Notice that there are four restrictions on $h(t)$: $h(1) = 1$ and $h(90) = 90$, since the time range of our temperature data is from Day 1 to Day 90 in each winter; $h(t_{1f}) = t_{0f}$, since the time warping function aligns x_1 at t_{1f} to x_0 at t_{0f}; $h(t)$ is strictly monotonic, with $s < t$ implies $h(s) < h(t)$. Following

these restrictions, the R function `landmarkreg` would use the linear interpolation method to estimate the time warping function between t_{0f} and t_{if} for all suitable f, using the following formula:

$$\frac{h(t) - t_{0f}}{t - t_{1f}} = \frac{t_{0f} - t_{0(f+1)}}{t_{1f} - t_{1(f+1)}}.$$

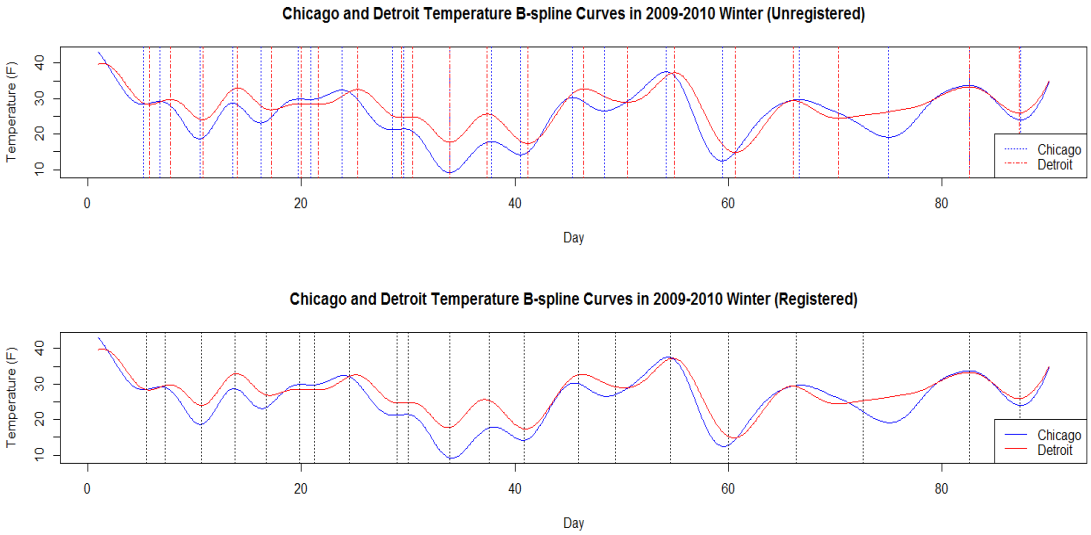

Fig. 4 The comparison between registered and unregistered B-spline curves of Chicago and Detroit daily temperature in 2009-2010 winter.

Figure 4 compares the landmark registration result for B-spline curves from the winter of 2009-2010. The dashed lines in the upper plot marks t_{0f} (Chicago, blue line) and t_{1f} (Detroit, red line). Obviously, the two B-spline curves have very similar pattern, and $t_{0f} < t_{1f}$ for nearly all f. This again justifies that there exists a time warping relationship between the daily temperatures in Chicago and Detroit in the winter months.

After applying landmark registration to the curves, as is shown in the lower plot, the dashed lines combined since $h(t_{1f}) = t_{0f}$ warped t_{0f} and t_{1f} together. The

slight shift of the curves in the lower plot comparing to the upper one indicates that the time lag between Chicago and Detroit is small.

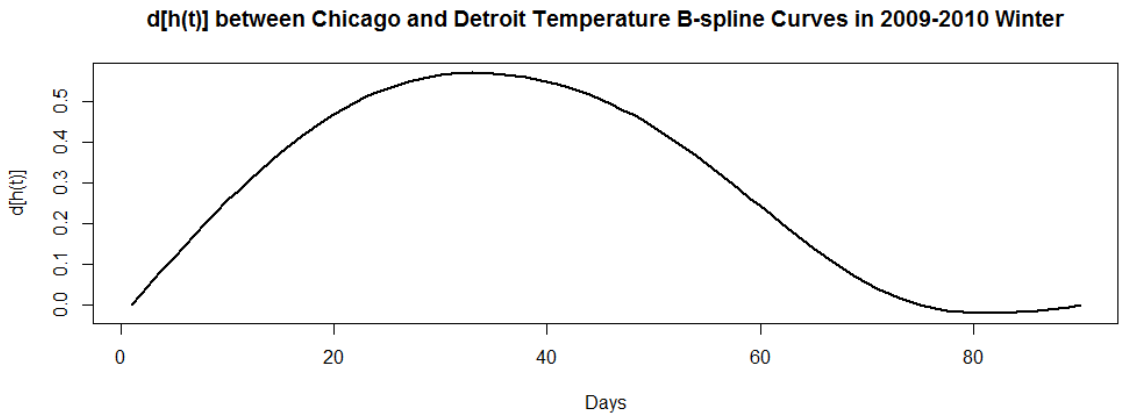

Fig. 5 The magnitude of $(t_{1f} - t_{0f})$ for Chicago and Detroit temperature B-spline curves in 2009-2010 winter.

As is shown in Figure 5, the time lag between Chicago and Detroit winter temperature in 2009-2010 is smaller than 0.6 days. Hence, by monitoring the temperature in Chicago in winter, we should be able to use this model and predict the temperature in Detroit within hours.

(4) Ordinary Least Squares Linear Model

Since pressure influences the speed of air flow, we built an ordinary least squares (OLS) linear model to estimate the temperature in Detroit using relevant meteorological data from both Chicago and Detroit, and compared to the estimated time lag with what we found using F.D.A. regression.

```
Residuals:
     Min       1Q   Median       3Q      Max
-0.33707 -0.17316 -0.00992  0.16144  0.34049

Coefficients:
                                  Estimate Std. Error t value Pr(>|t|)
(Intercept)                     -1.750271   3.092749  -0.566 0.572917
chicago_detroit_2009$chicago_STP  -0.023274   0.007101  -3.278 0.001511 **
chicago_detroit_2009$detroit_STP   0.025672   0.006359   4.037 0.000117 ***
chicago_detroit_2009$chicago_TEMP -0.013181   0.003470  -3.799 0.000270 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.195 on 86 degrees of freedom
Multiple R-squared:  0.2352, Adjusted R-squared:  0.2086
F-statistic: 8.818 on 3 and 86 DF,  p-value: 3.676e-05
```

Fig. 6 The summary of ordinary least squares (OLS) linear model using the temperature in Chicago and the station pressure in Chicago and Detroit to estimate the temperature in Detroit in the winter of 2009-2010.

```
> cbind(OLS,FDA)
           OLS       FDA
[1,] -194.7524 -193.9306
```

Fig. 7 The comparison of the sum of temperature differences between Chicago temperature at time t and Detroit temperature at time $(t+\delta)$, with the time lag δ estimated by OLS and F.D.A.

As is shown in Figure 6, OLS proves that the station air pressure in both cities and the temperature in Chicago are significant predictors for the temperature in Detroit. Furthermore, Figure 7 shows that in terms of the sum of error (in degrees Fahrenheit), the F.D.A. regression model has the same accuracy as the OLS model.

5, Results

The final comparison indicates that the F.D.A. model is equally accurate as the OLS model. In terms of the pros and cons, the amount of observed data that the OLS model requires is multiples of that needed by the F.D.A. approach, the latter saves much labor force and research funding. On the other hand, the F.D.A. model is constructed by functionalizing the true value of the response variable, in this paper is the Detroit winter temperature for a given year, while the OLS method regresses on influential parameters, the latter makes more sense for making predictions.

Nevertheless, the advantage of using the F.D.A. approach is obvious. We could control the trade-off between smoothness and accuracy in fitting curves by changing parameters such as `norder`, `nbasis`, and λ, etc. In addition, the functionalized curve provides more information on derivatives, therefore is more convenient for investigations relevant to rate, velocity, and acceleration, etc.

6, Acknowledgements

The author would like to express much thanks to Prof. Edward Rothman (University of Michigan – Ann Arbor) for supervising this research, and for patiently helping and proving advice as well as instructions for carrying out the data analysis and modeling. Gratitude is also extended to the viewers and editors for reading this manuscript. Additional thanks to the National Climatic Data Center for providing the data.

7, References

[1] Ramsay J, Hooker G, Graves S, "Functional Data Analysis with R and MATLAB", Springer Publishing Company, 2009

[2] Ramsay J, Silverman B, "Functional Data Analysis", Springer Publishing Company, 2005

[3] Ramsay J, Silverman B, "Applied functional data analysis: methods and case studies (Vol. 77)", New York: Springer, 2002

[4] Ramsay J, Dalzell C, "Some tools for functional data analysis", Journal of the Royal Statistical Society, Series B (Methodological), pp. 539-572, 1991

[5] Huld T, Šúri M, Dunlop E, Micale F, "Estimating average daytime and daily temperature profiles within Europe", Environmental Modelling & Software, 21(12), pp.1650-1661, 2006

8, Appendix of Code

(1) Data Input

```
########## Total data ##########

data = read.table("MI - 19361201-19660430.txt", header = TRUE, dec = ".")

data = data[with(data, order(data$YEARMODA)), ]

MItotal = data


data = read.table("MI - 19660501-19760430.txt", header = TRUE, dec = ".")

data = data[with(data, order(data$YEARMODA)), ]

MItotal = rbind(MItotal,data)


data = read.table("MI - 19760501-19820430.txt", header = TRUE, dec = ".")

data = data[with(data, order(data$YEARMODA)), ]

MItotal = rbind(MItotal,data)


data = read.table("MI - 19820501-19870430.txt", header = TRUE, dec = ".")

data = data[with(data, order(data$YEARMODA)), ]

MItotal = rbind(MItotal,data)


data = read.table("MI - 19870501-19910430.txt", header = TRUE, dec = ".")

data = data[with(data, order(data$YEARMODA)), ]

MItotal = rbind(MItotal,data)
```

```
data = read.table("MI - 19910501-19960430.txt", header = TRUE, dec = ".")

data = data[with(data, order(data$YEARMODA)), ]

MItotal = rbind(MItotal,data)


data = read.table("MI - 19960501-20010430.txt", header = TRUE, dec = ".")

data = data[with(data, order(data$YEARMODA)), ]

MItotal = rbind(MItotal,data)


data = read.table("MI - 20010501-20060430.txt", header = TRUE, dec = ".")

data = data[with(data, order(data$YEARMODA)), ]

MItotal = rbind(MItotal,data)


data = read.table("MI - 20060501-20100430.txt", header = TRUE, dec = ".")

data = data[with(data, order(data$YEARMODA)), ]

MItotal = rbind(MItotal,data)


data = read.table("MI - 20100501-20140430.txt", header = TRUE, dec = ".")

data = data[with(data, order(data$YEARMODA)), ]

MItotal = rbind(MItotal,data)


data = read.table("MI - 20140501-20150501.txt", header = TRUE, dec = ".")
```

```
data = data[with(data, order(data$YEARMODA)), ]

MItotal = rbind(MItotal,data)


write.csv(MItotal, "MItotal - 19361201-20150501.csv", quote=FALSE)

rm(data)


MItotal = read.csv("MItotal - 19361201-20150501.csv")

MIdaily = read.csv("MIdaily - 19361201-20150501.csv")


# Due to limited pages, Data Processing code is not shown in this paper
```

(2) B-spline Curve Fitting and Smoothing

```
# B-spline

plot(c(1:90), chicagonewmat[,9], type="o", lwd=2, xlab="Day", ylab="Temperature (F)", main="Chicago Winter Temperture and B-spline Curve in 1994"), par(mfrow=c(2,2))


tempbasis = create.bspline.basis(c(1,90),nbasis=20)

tempbasismat = eval.basis(c(1:90), tempbasis)

templist = smooth.basis(c(1:90), chicagonewmat[,9], tempbasis)

tempcoef = solve( crossprod(tempbasismat), crossprod(tempbasismat, chicagonewmat[,9]) )

tempfd = fd(tempcoef, tempbasis)

plot(tempfd, lty=1, lwd=2, col=1)

plotfit.fd(chicagonewmat[,9], c(1:90), tempfd[1], col="red", xlab="Day", ylab="Temperature (F)", main="Chicago Temperature and B-spline Curve in 2014-2015 Winter, with 20 basis functions")

points(c(1:90), chicagonewmat[,9], type="o", col="blue")

legend("topright", c("Temperature", "B-spline"), col=c("blue","red"), lty=c(1,1))


tempbasis = create.bspline.basis(c(1,90),nbasis=20)

tempbasismat = eval.basis(c(1:90), tempbasis)

templist = smooth.basis(c(1:90), detroitnewmat[,9], tempbasis)
```

```
tempcoef = solve( crossprod(tempbasismat), crossprod(tempbasismat, detroitnewmat[,9]))
tempfd = fd(tempcoef, tempbasis)
plot(tempfd, lty=1, lwd=2, col=1)
plotfit.fd(detroitnewmat[,9], c(1:90), tempfd[1], col="red", xlab="Day", ylab="Temperature (F)", main="Detroit Temperature and B-spline Curve in 2014-2015 Winter, with 20 basis functions")
points(c(1:90), detroitnewmat[,9], type="o", col="blue")
legend("topright", c("Temperature", "B-spline"), col=c("blue","red"), lty=c(1,1))

tempbasis = create.bspline.basis(c(1,90),nbasis=45)
tempbasismat = eval.basis(c(1:90), tempbasis)
templist = smooth.basis(c(1:90), chicagonewmat[,9], tempbasis)
tempcoef = solve( crossprod(tempbasismat), crossprod(tempbasismat, chicagonewmat[,9]))
tempfd = fd(tempcoef, tempbasis)
plot(tempfd, lty=1, lwd=2, col=1)
plotfit.fd(chicagonewmat[,9], c(1:90), tempfd[1], col="red", xlab="Day", ylab="Temperature (F)", main="Chicago Temperature and B-spline Curve in 2014-2015 Winter, with 45 basis functions")
points(c(1:90), chicagonewmat[,9], type="o", col="blue")
legend("topright", c("Temperature", "B-spline"), col=c("blue","red"), lty=c(1,1))
```

```
tempbasis = create.bspline.basis(c(1,90),nbasis=45)

tempbasismat = eval.basis(c(1:90), tempbasis)

templist = smooth.basis(c(1:90), detroitnewmat[,9], tempbasis)

tempcoef = solve( crossprod(tempbasismat), crossprod(tempbasismat, detroitnewmat[,9]))

tempfd = fd(tempcoef, tempbasis)

plot(tempfd, lty=1, lwd=2, col=1)

plotfit.fd(detroitnewmat[,9], c(1:90), tempfd[1], col="red", xlab="Day", ylab="Temperature (F)", main="Detroit Temperature and B-spline Curve in 2014-2015 Winter, with 45 basis functions")

points(c(1:90), detroitnewmat[,9], type="o", col="blue")

legend("topright", c("Temperature", "B-spline"), col=c("blue","red"), lty=c(1,1))
```

(3) Linear Models for Functional Responses

```
nbasis = 20

norder = 6

yearRng = c(1,90)


library(fda)

yearBasis = create.bspline.basis(yearRng, nbasis)


# choosing smoothing parameter lambda - GCV

#    set up a saturated basis: as many basis functions as observations

dayrange = c(1,90); dayTime = 1:90

daybasis = create.fourier.basis(dayrange, nbasis)


#    set up the harmonic acceleration operator

Lcoef              = c(0,(2*pi/diff(dayrange))^2,0)

harmaccelLfd = vec2Lfd(Lcoef, dayrange)


#    step through values of log(lambda)


loglam             = seq(0.379,0.382,0.0001)

nlam                = length(loglam)

dfsave             = rep(NA,nlam)
```

```
names(dfsave) = loglam
gcvsave             = dfsave
for (ilam in 1:nlam) {
   cat(paste('log10 lambda =',loglam[ilam],'\n'))
   lambda               = 10^loglam[ilam]
   fdParobj            = fdPar(daybasis, harmaccelLfd, lambda)
   smoothlist        = smooth.basis((0.5+0:89), chicagonewmat[,9], fdParobj)
   dfsave[ilam]    = smoothlist$df
   gcvsave[ilam] = sum(smoothlist$gcv)
}
plot(loglam, gcvsave, type='b', lwd=2, ylab='GCV Criterion',
        xlab=expression(log[10](lambda)) )

# use the GCV lambda = 10^(0.685) on fdPar for detroitnewmat[,9]
D2fdPar = fdPar(yearBasis, lambda=10^(1.65))
detroitfd = smooth.basis(dayTime, detroitnewmat[,9], D2fdPar)$fd

# use the GCV lambda = 10^(0.685) on fdPar for chicagonewmat[,9]
D2fdPar = fdPar(yearBasis, lambda=10^(0.38))
chicagofd = smooth.basis(dayTime, chicagonewmat[,9], D2fdPar)$fd
```

```
# for each of 144 birth year cohorts

plotfit.fd(detroitnewmat[,9],dayTime,detroitfd,col="red")

points(c(1:90), detroitnewmat[,9], type="o", col="blue")

legend("topright", c("Temperature", "B-spline"), col=c("blue","red"), lty=c(1,1))


plotfit.fd(chicagonewmat[,9],dayTime,chicagofd)


# Set up for the list of regression coefficient fdPar objects


tempBetaBasis = create.bspline.basis(yearRng,nbasis)


tempBeta0Par = fdPar(tempBetaBasis, 2)


tempBeta1fd    = bifd(matrix(0,20,20), tempBetaBasis, tempBetaBasis)


tempBeta1Par = bifdPar(tempBeta1fd, 2, 2,5000,5000)


tempBetaList = list(tempBeta0Par, tempBeta1Par, tempBeta1Par)


#    Define the dependent and independent variable objects


Next = detroitfd
```

```
Last = chicagofd


#    Do the regression analysis

# override function linmod using linmod.R

temp.linmod = linmod(Last, Next, tempBetaList)


temp.days = seq(1, 90, 1) # original: seq(1, 365, 8)

temp.beta0mat = eval.fd(temp.days, temp.linmod$beta0estfd)

temp.beta1mat = eval.bifd(temp.days, temp.days, temp.linmod$beta1estbifd)


# temp.beta1mat +/- 1~2

temp.beta1matp1 = eval.bifd(c(2:90), c(1:89), temp.linmod$beta1estbifd)

temp.beta1matm1 = eval.bifd(c(1:89), c(2:90), temp.linmod$beta1estbifd)

temp.beta1matp2 = eval.bifd(c(3:90), c(1:88), temp.linmod$beta1estbifd)

temp.beta1matm2 = eval.bifd(c(1:88), c(3:90), temp.linmod$beta1estbifd)


# Figure 10.11

nrz <- length(temp.days)

ncz <- length(temp.days)

jet.colors <- colorRampPalette( c("floralwhite", "midnightblue") )

nbcol <- 100

color <- jet.colors(nbcol)
```

```
zfacet <- temp.beta1mat[-1, -1] + temp.beta1mat[-1, -ncz] +
   temp.beta1mat[-nrz, -1] + temp.beta1mat[-nrz, -ncz]
# Recode facet z-values into color indices
facetcol <- cut(zfacet, nbcol)

# change temp.days = seq(1, 365, 8) before plotting
persp(temp.days, temp.days, temp.beta1mat, xlab="Days",
ylab="Days",zlab="beta(s,t)",
        cex.lab=1.2,cex.axis=2, phi=25, theta=-30,expand=1,
        border="black",col=color[facetcol])
legend("bottomright","z-axis not shrinked")

library(latex2exp)
plot(temp.beta0mat,xlab="Days",ylab=latex2exp('$\\alpha$(t)'),type='o',
      main=latex2exp('Detroit ~ Chicago 1973-2014 functional regression
$\\alpha$(t)'))

plot(diag(temp.beta1mat),xlab="Days",ylab=latex2exp('$\\beta$'),type="l",col=1,
      main=latex2exp('Detroit ~ Chicago 1973-2014 functional regression $\\beta$'))
lines(y=diag(temp.beta1matp1),x=c(1:89),type="l",col=2)
lines(y=diag(temp.beta1matm1),x=c(2:90),type="l",col=3)
lines(y=diag(temp.beta1matp2),x=c(1:88),type="l",col=4)
```

```
lines(y=diag(temp.beta1matm2),x=c(3:90),type="l",col=5)
legend("top",c(latex2exp('$\\beta$(t, t)'), latex2exp('$\\beta$(t+1, t)'),
                    latex2exp('$\\beta$(t-1, t)'), latex2exp('$\\beta$(t+2, t)'),
                    latex2exp('$\\beta$(t-2, t)')),
       lty=1, col=c(1,2,3,4,5))

# Data processing
detroitnewtotal = MItotal[(which(MItotal$STN...==725370)),]
detroittotal = detroitnewtotal
rm(detroitnewtotal)

data = detroittotal[,c("YEARMODA","TEMP","SLP","STP","MAX","MIN")]
which(data$YEARMODA == 19731201)    # = 1850
data = data[c(1850:nrow(data)),]
summary(data)

dat = data[0,]
for (i in 1:nrow(data)) {
   print(i)
   n = data[i,1]%%10000
   if (n < 229 || n > 1200) {
      dat = rbind(dat,data[i,])
```

```
    }
}
rm(n,i,j)


data = dat; rm(dat); row.names(data) = NULL
length(which(data$STP==9999.9))


data[,c(5,6)] = lapply(data[,c(5,6)], as.character)
for (i in 1:nrow(data)) {
    for (j in 5:6) {
        print(i)
        if (substr(data[i,j], nchar(data[i,j]), nchar(data[i,j])) < '0' || substr(data[i,j],
nchar(data[i,j]), nchar(data[i,j])) > '9') {
            data[i,j] = substr(data[i,j], 1, nchar(data[i,j])-1) }
    }
}
data[,c(5,6)] = lapply(data[,c(5,6)], as.numeric)
data = data[with(data, order(YEARMODA)), ]


detroit = data
rm(data)
detroit$DTR = detroit$MAX - detroit$MIN
```

(4) Landmark Registration

```
plot(chicagomat,col=c(1:42))

matplot(c(1:90),chicagomat[,c(1:5)],type="l")


nbasis = 65

yearBasis = create.bspline.basis(norder=6,breaks=c(1:90))


#    comsider = 3 for penalize curvature of acceleration

Lfdobj       = c(0,(2*pi/diff(dayrange))^2,0)


detroitlambda       = 10^(1.7)    #    smoothing parameter

chicagolambda       = 10^(0.327)    #    smoothing parameter


detroitvecf         = matrix(0, yearBasis$nbasis, ncol(detroitmat))

chicagovecf         = matrix(0, yearBasis$nbasis, ncol(chicagomat))


dimnames(detroitvecf) = list(yearBasis$names, dimnames(detroitmat)[[2]])

dimnames(chicagovecf) = list(yearBasis$names, dimnames(chicagomat)[[2]])


detroitfd0      = fd(detroitvecf, yearBasis)

detroitfdPar = fdPar(detroitfd0, Lfdobj, detroitlambda)

chicagofd0      = fd(chicagovecf, yearBasis)
```

```
chicagofdPar = fdPar(chicagofd0, Lfdobj, chicagolambda)

detroitsmooth = smooth.basis(c(1:90), detroitmat, detroitfdPar)
detroitfd = detroitsmooth$fd
chicagosmooth = smooth.basis(c(1:90), chicagomat, chicagofdPar)
chicagofd = chicagosmooth$fd

detroitaccelfd          = deriv.fd(detroitfd, 2)
detroitaccelmeanfd = mean(detroitaccelfd)

chicagoaccelfd          = deriv.fd(chicagofd, 2)
chicagoaccelmeanfd = mean(chicagoaccelfd)

plot(detroitaccelfd, lty=1, lwd=2, xlab="Day", ylab="Acceleration (F/day^2)",
        main="Detroit unregistered curves")
plot(detroitaccelfd[seq(1,42,by=2)], lty=1, lwd=2, xlab="Day", ylab="Acceleration
(F/day^2)",
        main="Detroit unregistered curves (1973, 1977, 1981, ..., 2009, 2013)")

plot(chicagoaccelfd, lty=1, lwd=2, xlab="Day", ylab="Acceleration (F/day^2)",
        main="Chicago unregistered curves")
plot(chicagoaccelfd[seq(1,42,by=4)], lty=1, lwd=2, xlab="Day", ylab="Acceleration
```

```
(F/day^2)",
        main="Chicago unregistered curves (1973, 1977, 1981, ..., 2009, 2013)")

# try out other variables besides TEMP
detroitmat = matrix(detroit[,2],nrow=90,byrow=F)
dimnames(detroitmat)[[2]] <- paste(1973:2014, sep='')

detroitSLPmat = matrix(detroit[,3],nrow=90,byrow=F)
dimnames(detroitSLPmat)[[2]] <- paste(1973:2014, sep='')

detroitMAXmat = matrix(detroit[,5],nrow=90,byrow=F)
dimnames(detroitMAXmat)[[2]] <- paste(1973:2014, sep='')

detroitMINmat = matrix(detroit[,6],nrow=90,byrow=F)
dimnames(detroitMINmat)[[2]] <- paste(1973:2014, sep='')

detroitDTRmat = matrix(detroit[,7],nrow=90,byrow=F)
dimnames(detroitDTRmat)[[2]] <- paste(1973:2014, sep='')

########################## sample years ##########################
sampleyear = seq(1974,2014,by=5)    # 1974 1979 1984 1989 1994 1999 2004 2009
2014
```

```
sampleyearcol = seq(2,42,by=5)

# colnames(chicagomat)[sampleyearcol] == sampleyear

chicagonewmat = chicagomat[,sampleyearcol]

detroitnewmat = detroitmat[,sampleyearcol]

tempnewmat = list(cbind(chicagonewmat[,1],detroitnewmat[,1]),
                        cbind(chicagonewmat[,2],detroitnewmat[,2]),
                        cbind(chicagonewmat[,3],detroitnewmat[,3]),
                        cbind(chicagonewmat[,4],detroitnewmat[,4]),
                        cbind(chicagonewmat[,5],detroitnewmat[,5]),
                        cbind(chicagonewmat[,6],detroitnewmat[,6]),
                        cbind(chicagonewmat[,7],detroitnewmat[,7]),
                        cbind(chicagonewmat[,8],detroitnewmat[,8]),
                        cbind(chicagonewmat[,9],detroitnewmat[,9]))


# registration

matplot(c(1:90),tempnewmat[[9]],type="l",
        xlab="Day", ylab="Daily mean temperature",
        main="2014 winter daily mean temp")


nbasis = 65

yearBasis = create.bspline.basis(norder=6,breaks=c(1:90))
```

```
#    comsider = 3 for penalize curvature of acceleration
# Lfdobj       = c(0,(2*pi/diff(dayrange))^2,0)
Lfdobj       = 3

loglam              = seq(2.27,2.28,0.001)
nlam                = length(loglam)
dfsave              = rep(NA,nlam)
names(dfsave) = loglam
gcvsave             = dfsave
for (ilam in 1:nlam) {
   cat(paste('log10 lambda =',loglam[ilam],'\n'))
   lambda              = 10^loglam[ilam]
   fdParobj            = fdPar(daybasis, harmaccelLfd, lambda)
   smoothlist       = smooth.basis((0.5+0:89), tempnewmat[[4]], fdParobj)
   dfsave[ilam]    = smoothlist$df
   gcvsave[ilam] = sum(smoothlist$gcv)
}
plot(loglam, gcvsave, type='b', lwd=2, ylab='GCV Criterion',
        xlab=expression(log[10](lambda)) )

#    smoothing parameter
detroitnewmatlambda = 10^c(2.2, 3.12, 2.29, 3.36, 0.51, 2.69, 4.01, 0.043, 3.19)
```

```
chicagonewmatlambda = 10^c(2.42, 3.1, 2.26, 3.46, 2.648, 2.22, 3.45, 0.085, 3.049)
# tempnewlambda = list(cbind(chicagonewmatlambda[1],detroitnewmatlambda[1]),
#
cbind(chicagonewmatlambda[2],detroitnewmatlambda[2]),
#
cbind(chicagonewmatlambda[3],detroitnewmatlambda[3]),
#
cbind(chicagonewmatlambda[4],detroitnewmatlambda[4]),
#
cbind(chicagonewmatlambda[5],detroitnewmatlambda[5]),
#
cbind(chicagonewmatlambda[6],detroitnewmatlambda[6]),
#
cbind(chicagonewmatlambda[7],detroitnewmatlambda[7]),
#
cbind(chicagonewmatlambda[8],detroitnewmatlambda[8]),
#
cbind(chicagonewmatlambda[9],detroitnewmatlambda[9]))
tempnewlambda = (detroitnewmatlambda + chicagonewmatlambda) / 2


detroitnewvecf          = matrix(0, yearBasis$nbasis, 1)
chicagonewvecf          = matrix(0, yearBasis$nbasis, 1)
```

```
tempnewvecf             = matrix(0, yearBasis$nbasis, 2) # run for each element in
tempnewmat


# now process: 2009
dimnames(detroitnewvecf) = list(yearBasis$names, sampleyear[1])
dimnames(chicagonewvecf) = list(yearBasis$names, sampleyear[1])
dimnames(tempnewvecf)    = list(yearBasis$names, c("Chicago", "Detroit"))


detroitnewfd0   = fd(detroitnewvecf, yearBasis)
detroitnewfdPar = fdPar(detroitnewfd0, Lfdobj, detroitnewmatlambda[1])
chicagonewfd0   = fd(chicagonewvecf, yearBasis)
chicagonewfdPar = fdPar(chicagonewfd0, Lfdobj, chicagonewmatlambda[1])
tempnewfd0      = fd(tempnewvecf, yearBasis)
tempnewfdPar = fdPar(tempnewfd0, Lfdobj, tempnewlambda[8])


detroitnewsmooth = smooth.basis(c(1:90), detroitnewmat[,1], detroitnewfdPar)
detroitnewfd = detroitnewsmooth$fd
chicagonewsmooth = smooth.basis(c(1:90), chicagonewmat[,1], chicagonewfdPar)
chicagonewfd = chicagonewsmooth$fd
tempnewsmooth = smooth.basis(c(1:90), tempnewmat[[8]], tempnewfdPar)
tempnewfd = tempnewsmooth$fd
```

```
# detroitnewaccelfd          = deriv.fd(detroitnewfd, 2)
# detroitnewaccelmeanfd = mean(detroitnewaccelfd)
# chicagonewaccelfd          = deriv.fd(chicagonewfd, 2)
# chicagonewaccelmeanfd = mean(chicagonewaccelfd)
# tempnewaccelfd          = deriv.fd(tempnewfd, 2)
# tempnewaccelmeanfd = mean(tempnewaccelfd)

plot(detroitnewaccelfd, lty=1, lwd=2, xlab="Day", ylab="Acceleration (F/day^2)",
        main="Detroit unregistered curves")
plot(chicagonewaccelfd, lty=1, lwd=2, xlab="Day", ylab="Acceleration (F/day^2)",
        main="Chicago unregistered curves")

# legend("bottomleft",c("Chicago", "Detroit"), lty=1, col=c(1,2))
plot(tempnewfd,Lfdobj=1)

# PGS spurt identification
# index    = 1:600    #    wide limits
# nindex = length(index)
# ageval = seq(15,75,len=600)
# ncasef = 2
# PGSctr = rep(0,ncasef)
# op = par(mfrow=c(2,1))
```

```
# for (icase in 1:ncasef) {
#     accveci = eval.fd(ageval, tempnewfd[icase])
#     aup         = accveci[2:nindex] - 25
#     adn         = accveci[1:(nindex-1)] - 25
#     indx        = (1:600)[adn*aup < 0 & adn > 0]
#     plot(ageval[2:nindex],aup,"p")
#     lines(c(0,90),c(25,25),lty=2)
#     for (j in 1:length(indx)) {
#       indxj = indx[j]
#       aupj    = aup[indxj]
#       adnj    = adn[indxj]
#       agej    = ageval[indxj] + 0.1*(adnj/(adnj-aupj))
#       if (j == length(indx)) {
#         PGSctr[icase] = agej
#         lines(c(agej,agej),c(-15,15),lty=1)
#         lines(c(15,75),c(0,0),lty=3)
#       } else {
#         lines(c(agej,agej),c(-15,15),lty=3)
#         lines(c(15,75),c(0,0),lty=3)
#       }
#     }
#     title(paste('City ',c("Chicago", "Detroit")[icase]))
```

```
# }
# par(op)

cities = 1:2
landmarkpts = 21
nfine      = 90
dayfine = seq(1, 90, length=nfine)

# use 1st order derivatives to find landmark points
TEMPctr = matrix(0,nrow=length(cities), ncol=landmarkpts)
for (icase in cities) {
   TEMPveci = eval.fd(dayfine, tempnewfd[icase])
   plot(tempnewfd,Lfdobj=1)

   for (ptscase in 1:landmarkpts) {
      TEMPctr[icase, ptscase] = locator(1)$x
   }
}

TEMPctrmean = mean(TEMPctr)

# plot unregistrated curves
```

```
plot(tempnewfd, lty=1, xlab="Day", ylab="Temperature (F)", col=c("blue","red"),
        main="Chicago and Detroit Temperature B-spline Curves in 2004-2005 Winter
(Unregistered)")
abline(v =TEMPctr[1,], col="blue", lty=3)
abline(v =TEMPctr[2,], col="red", lty=4)
legend("topright", c("Chicago", "Detroit"), col=c("blue","red"), lty=c(3,4))

#    Define the basis for the function W(t).
wbasisLM        =        create.bspline.basis(c(1,90),        nbasis=27,        norder=6,
c(1,colMeans(TEMPctr),90)) # nbasis = norder + length(breaks) - 2
WfdLM       = fd(matrix(0,27,1),wbasisLM)
WfdParLM = fdPar(WfdLM,1,1e3)

#    Carry out landmark registration.

regListLM  =  landmarkreg(tempnewfd,  TEMPctr,  colMeans(TEMPctr),  WfdParLM,
TRUE) # Detroit -> Chicago

accelfdLM          = regListLM$regfd
accelmeanfdLM = mean(accelfdLM)

#    plot registered curves
```

```
plot(accelfdLM, lty=1, lwd=1, main="2009 winter 3 Months Landmark registration",
        cex=2, xlab="Day", ylab="Temperature (F)")
abline(v =colMeans(TEMPctr), col=1, lty=3)

lines(accelmeanfdLM, col=1, lwd=2, lty=2)
lines(c(TEMPctrmean,TEMPctrmean), c(10,40), lty=2, lwd=1.5)
lines(c(0,90), c(25,25), lty=2, lwd=1.5)

plot(accelfdLM[1:2], lty=1, lwd=1,
        cex=2, xlab="", ylab="Acceleration (cm/yr/yr)")
lines(mean(accelfdLM), col=1, lwd=2, lty=2)
lines(c(TEMPctrmean,TEMPctrmean), c(-3,1.5), lty=2, lwd=1.5)

#    plot warping functions
warpfdLM    = regListLM$warpfd
warpmatLM = eval.fd(dayfine, warpfdLM)
warpmatLM = as.data.frame(warpmatLM)
warpmatLM$Mean = apply(warpmatLM,1,mean)

plot(tempnewfd[1], lty=1, lwd=2,
        xlab="", ylab="")
```

```
lines(c(TEMPctrmean,TEMPctrmean), c(1,90), lty=2, lwd=1.5)
plot(dayfine, warpmatLM[,2], "l", lty=1, lwd=2, col=1, cex=1.2,
     xlab="Days", ylab="Time warping function h(t)",main="2009 Winter: Dec-Feb")
lines(dayfine,  dayfine, lty=2, lwd=1.5,col=2)
lines(c(TEMPctrmean,TEMPctrmean), c(1,90), lty=2, lwd=1.5)

plot(dayfine, warpmatLM[,2]-warpmatLM[,1], "l", lty=1, lwd=2, col=1, cex=1.2,
     xlab="Days", ylab="d[h(t)]",main="2009 Winter: Dec-Feb d[h(t)]")
text(TEMPctrmean+0.1, warpmatLM[61,3]+0.3, "o", lwd=2)

# delta h(t) and delta pressure
dPressure = chicago$STP - detroit$STP
chicago_detroit_2009$dPressure_SLP = chicago$SLP[3241:3330] - detroit$SLP[3241:3330]
chicago_detroit_2009$dPressure_STP = chicago$STP[3241:3330] - detroit$STP[3241:3330]

chicago_detroit_2009$detroit_STP = detroit$STP[3241:3330]
chicago_detroit_2009$chicago_STP = chicago$STP[3241:3330]

chicago_detroit_2009$detroit_SLP = detroit$SLP[3241:3330]
```

```
chicago_detroit_2009$chicago_SLP = chicago$SLP[3241:3330]


# Year 2004: row 2701-2790
# dWarp = (warpmatLM[,2]-warpmatLM[,1])[seq(1,500,length.out=90)]
dWarp = warpmatLM[,2]-warpmatLM[,1]
chicago_detroit_2009$dWarp = warpmatLM[,2]-warpmatLM[,1]
chicago_detroit_2009_select                                                    =
chicago_detroit_2009[which(chicago_detroit_2009$dWarp > 0),]
```

(5) Ordinary Least Squares Linear Model

```
reg09=glm(chicago_detroit_2009_select$dWarp~chicago_detroit_2009_select$dPress
ure, family=gaussian())
reg09=lm(chicago_detroit_2009_select$dWarp~chicago_detroit_2009_select$dPressu
re)
summary(reg09)
plot(chicago_detroit_2009_select$dPressure,      chicago_detroit_2009_select$dWarp,
main="2009 Winter: Dec-Feb dWarp~dPressure (STP)")
abline(reg04)

# consider inverse.gaussian and poisson for glm
chicago_detroit_2009$dTemp=chicago_detroit_2009$chicago_TEMP-chicago_detroit
_2009$detroit_TEMP
reg09 = glm(chicago_detroit_2009$dWarp~
                    chicago_detroit_2009$chicago_STP
                  +chicago_detroit_2009$detroit_STP
#                   +chicago_detroit_2009$chicago_SLP
#                   +chicago_detroit_2009$detroit_SLP
                    +chicago_detroit_2009$chicago_TEMP
#                  +chicago_detroit_2009$detroit_TEMP
                  , family=gaussian())
```

```
reg09 = lm((chicago_detroit_2009$dWarp)[2:89]~
                    (chicago_detroit_2009$chicago_STP)[2:89]
                  +(chicago_detroit_2009$detroit_STP)[2:89]
                  +(chicago_detroit_2009$chicago_TEMP)[2:89])

summary(reg09)
attach(chicago_detroit_2009)
chicago_detroit_2009$predict = predict(reg09, chicago_detroit_2009[,c(2,3,5)])
chicago_detroit_2009$predict_diff = chicago_detroit_2009$predict -
chicago_detroit_2009$dWarp
sum(predict_diff*24 < 3) / length(predict_diff)

reg09 = glm(chicago_detroit_2009$dWarp~chicago_detroit_2009$detroit_STP,
family=gaussian())
reg09 =
lm(chicago_detroit_2009$dWarp~chicago_detroit_2009$detroit_STP+chicago_detroi
t_2009$detroit_STP)

plot(chicago_detroit_2009$dWarp, chicago_detroit_2009$detroit_STP, main="2009
Winter: Dec-Feb dWarp~Detroit_STP")
abline(reg04)
```

```
# cont. reg

nwbasisCR = 65

norderCR    =    6

wbasisCR    = create.bspline.basis(c(1,90), nwbasisCR, norderCR)

Wfd0CR        = fd(matrix(0,nwbasisCR,2),wbasisCR)

lambdaCR    = tempnewlambda[1]

WfdParCR    = fdPar(Wfd0CR, 1, lambdaCR)


#    carry out the registration


registerlistCR = register.fd(mean(tempnewfd), tempnewfd, WfdParCR)


accelfdCR = registerlistCR$regfd

WfdCR          = registerlistCR$Wfd


#    plot landmark and continuously registered curves for the

#    first 10 children


accelmeanfdLM74 = mean(tempnewfd[c(1:2)])

accelmeanfdCR74 = mean(tempnewfd[c(1:2)])


plot(tempnewfd[c(1:2)], lty=1, lwd=1,main="Continuous registration",
```

```
        cex=2, xlab="Day", ylab="Temperature (F)")
lines(accelmeanfdLM74, col=1, lwd=2, lty=2)
lines(c(TEMPctrmean,TEMPctrmean), c(10,40), lty=2, lwd=1.5)
lines(c(0,90), c(25,25), lty=2, lwd=1.5)



# dPressure dig deep
chicago_detroit_STP = as.data.frame(cbind(detroit$YEARMODA, detroit$STP,
chicago$STP))
colnames(chicago_detroit_STP) = c("YEARMODA", "detroit_STP", "chicago_STP")
# take STP in chicago - STP in detroit as dPressure
chicago_detroit_STP$dPressure = chicago_detroit_STP$chicago_STP -
chicago_detroit_STP$detroit_STP
View(chicago_detroit_STP[which(chicago_detroit_STP$chicago_STP==9999.9 |
chicago_detroit_STP$detroit_STP==9999.9),])
# missing STP for 2004 2003.1-2 2002 2001 1999


# Hence, use Year 2009 as an example
chicago_detroit_2009 =
chicago_detroit_STP[which(chicago_detroit_STP$YEARMODA > 20091200
                                                    &
chicago_detroit_STP$YEARMODA < 20100300),]
```

```
row.names(chicago_detroit_2009) = NULL

chicago_detroit_2009$chicago_TEMP = chicagonewmat[,"2009"]

chicago_detroit_2009$detroit_TEMP = detroitnewmat[,"2009"]


# predict_diff = warpmatLM[,2]-warpmatLM[,1]


output1 = cbind(predict,predict*24,predict_diff,predict_diff*24)

colnames(output1) = c("OLS time lag (days)", "OLS time lag (hours)",

                      "F.D.A. time lag (days)", "F.D.A. time lag (hours)")

View(output1)


timeols = c(1:90)-predict

(OLS = sum(eval.fd(timeols[2:89],tempnewfd[1])-eval.fd(c(2:89),tempnewfd[2])))

(FDA=sum(eval.fd(warpmatLM[2:89,1],tempnewfd[1])-eval.fd(warpmatLM[2:89,2],t

empnewfd[2])))

cbind(OLS,FDA)
```