British Journal of Mathematics & Computer Science
7(5): 341-357, 2015, Article no.BJMCS.2015.129
ISSN: 2231-0851

SCIENCEDOMAIN *international*
www.sciencedomain.org

Breeding Value Prediction Using a Functional Data Multiple Regression Equation

Kunio Takezawa[1*]

[1] *Agroinformatics Division, Agricultural Research Center, National Agriculture and Food Research Organization, Kannondai 3-1-1, Tsukuba, Ibaraki 305-8666, Japan.*

Article Information
DOI: 10.9734/BJMCS/2015/16480
Editor(s):
(1) Sheng Zhang, Department of Mathematics, Bohai University, Jinzhou, China.
Reviewers:
(1) Klra Kosov, Division of Plant Genetics and Breeding, Laboratory of Plant Stress Biology and Biotechnology, Crop Research Institute, Prague, Czech Republic.
(2) Edward Missanjo, Malawi College of Forestry and Wildlife, Private Bag 6, Dedza, Malawi.
Complete Peer review History:
http://www.sciencedomain.org/review-history.php?iid=936&id=6&aid=8218

Original Research Article

Received: 03 February 2015
Accepted: 16 February 2015
Published: 24 February 2015

Abstract

In this study, the applicability of a multiple regression equation to predict breeding values based on the high-density SNP (single nucleotide polymorphism) markers that are found in the whole genome sequences of animals and plants was evaluated. The genotypes of a large number of SNPs distributed on chromosomes were treated as functional data and phenotypic values of a trait were treated as scalar target variables in the functional data multiple regression equations. The functional data analysis R package ("fda", version 2.4.0) was used to create the functional data multiple linear regression equations. An outline of this procedure is presented in this paper. We evaluated the accuracy of the functional data multiple regression equations by predicting breeding values using simulated data sets of SNPs as predictors and phenotypic values of a trait as variables. We found that the regression equations predicted the breeding values with considerable accuracy even though the predictors were not selected, nor were prior distributions assumed.

Keywords: B-spline; genome sequence; genomic selection; genotype; single nucleotide polymorphism; smoothing splines.
2010 Mathematics Subject Classification: 62E17; 62F10; 62J99

**Corresponding author: E-mail: nonpara@gmail.com*

1 Introduction

Methods and instruments for analysis of DNA (deoxyribonucleic acid) sequence data have progressed rapidly, and enormous amounts of genomic information for many species, including humans, have accumulated as a result [1]. Genomic information of crop plants such as rice and wheat and of livestock such as cow and pig has also become available. Among the genomic information, mutations in the DNA sequences, which characterize individuals, breeds of livestock, and varieties and inbred lines of crop plants are called DNA markers. SNPs (single nucleotide polymorphisms) are DNA markers that are densely distributed in genomes. The many thousands of SNPs can be used to study generic variations in whole genomes. SNPs have been used to predict genetic abilities that determine, for example, quantities of agricultural products such as yield amounts and meat productivities. Such predictions have contributed to the establishment a new efficient breeding method called genomic selection for selecting individuals and breed varieties with beneficial properties [2].

The value of an individual plant or an animal as a genetic parent is called its breeding value. The accurate prediction of breeding values of individuals or breeds using large SNP genotype data sets plays an important role in genomic selection. In general, two different alleles exist at a SNP locus. Combinations of these two alleles give three types of SNP genotypes: homozygous with one of the allele, homozygous with the other allele, and heterozygous. For instance, when the two alleles are A (adenine) and T (thymine), the three possible SNP genotypes are AA, AT, and TT. These SNP genotypes are coded as 1 and -1 for two homozygous and 0 for heterozygous to be used as predictors in prediction models.

Although our goal is the prediction of breeding values, the target variable of the prediction model is the phenotypic value of a trait. While the numbers of SNPs in a genome ranges from a few thousands to a few hundreds of thousands, the numbers of phenotypes of individuals or breeds usually ranges from several dozens to a few thousands; therefore, we have to create prediction models when the number of predictors (SNPs) is much larger than the number of variables (phenotypes). A variety of methods have been employed to create breeding value prediction models: linear regressions using regularization methods such as LASSO, ridge regression, and Bayesian methods; and prediction models using machine learning techniques such as kernel regression [3-8]. All these techniques are dimension reduction methods that use selection of predictors or shrinkage regression methods. However, because SNPs are arranged densely on a chromosome, neighboring SNPs are highly correlated because of linkage disequilibrium. If a chromosome is regarded as a time sequence, the genotypes of a large number of SNPs can be treated as functional data that represent variation in time. Because the numbers of available SNPs are likely to increase in the near future, SNPs arranged on a chromosome are expected to be considered a continuous variable. Therefore, the possibility of treating SNPs as functional data in breeding value prediction models should be considered. In this paper, we investigate the application of functional data analysis for developing breeding value prediction models aimed at genomic selection.

In a regression equation of breeding value prediction models, we treat SNP data as predictors (functional data) and the numerical values as the target variable (scalar). In this form, the regression equation is the basic regression equation that has been used widely in functional data analysis [9,10], section 5 of [11,12].

The functional data analysis R package ("fda", version 2.4.0)[13] is a publicly available tool for implementing functional data analysis using the R language. This package allows diverse regressions using functional data analysis and related statistical computation to be carried out efficiently and in a sophisticated manner. A user manual for the package "fda" (version 2.4.0) tool is freely available [13].

The simplest regression equation in which predictors are functional data and the target variable is scalar is a regression equation with one functional data set as a predictor ($\{x_i(t)\}(1 \leq i \leq n)$ (n is the number of items in the data set) and one target variable data set ($\{y_i\}(1 \leq i \leq n)$). The relationship between the predictor and the target variable is represented as has been described previously (e.g.,

see page 261 in [10] and page 133 in [12])

$$y_i = \alpha_0 + \int_{t^{min}}^{t^{max}} x_i(t)\beta(t)dt + \epsilon_i, \tag{1.1}$$

where α_0 is a constant, and t^{min} and t^{max} are the minimum and maximum of the range of t; t is the argument of $\{x_i(t)\}$. $\beta(t)$ plays the same role as the regression coefficients in a usual regression equation. $\{\epsilon_i\}(1 \leq i \leq n)$ defines the residuals. When both $x_i(t)$ and $\beta(t)$ are scalars, this regression is equivalent to a simple regression. Hence, this regression is regarded as an extension of a simple regression using functional data as predictors.

Therefore, generalization of Eq.(1.1) yields a multiple linear regression equation with K predictors. That is,

$$y_i = \alpha_0 + \sum_{k=1}^{K} \int_{t_k^{min}}^{t_k^{max}} x_{ik}(t)\beta_k(t)dt + \epsilon_i, \tag{1.2}$$

where the predictors are functions ($\{x_{ik}(t)\}(1 \leq i \leq n, 1 \leq k \leq K)$), and the target variable is a scalar ($\{y_i\}(1 \leq i \leq n)$). $\beta_k(t)$ corresponds to the regression coefficient of the k-th predictor in a usual multiple linear regression. α_0 is a constant, t_k^{min} and t_k^{max} are the minimum and maximum of the range of t, and the argument of $\{x_{ik}(t)\}$; $\{x_{ik}(t)\}$ corresponds to the k-th predictors in a usual multiple regression. $\{\epsilon_i\}(1 \leq i \leq n)$ indicates residuals.

The functional data multiple linear regression equation is also considered a generalization of an additive model, which becomes obvious when $\{x_{ik}(t)\}$ in Eq.(1.2) is written as

$$x_{ik}(t) = \delta(t - \tau_{ik}), \tag{1.3}$$

where $\delta(\cdot)$ is a delta function. This setting transforms Eq.(1.2) to

$$\begin{aligned} y_i &= \alpha_0 + \sum_{k=1}^{K} \int_{t_k^{min}}^{t_k^{max}} \delta(t - \tau_{ik})\beta_k(t)dt + \epsilon_i \\ &= \alpha_0 + \sum_{k=1}^{K} \beta_k(\tau_{ik}) + \epsilon_i. \end{aligned} \tag{1.4}$$

This equation describes an additive model in which $\{\tau_{ik}\}$ plays the role of predictors of data. Therefore, the additive model can be regarded as a special case of the functional data multiple regression equation.

Package “fda” can be used to produce a regression equation in the form of Eq.(1.2) in a simple way. However, in their paper Ramsay et al.[12] focused almost exclusively on a regression equation with one predictor and $\{x_i(t)\}$ that were assumed to be a periodic functions. Moreover, the equations behind the commands in package “fda” were not explained in detail. Here, we first explain the procedures for creating regression equations in the form of Eq.(1.2) and give a general outline of the calculations in these processes using simple examples with two predictors and the nonperiodic function $\{x_i(t)\}$. We also describe a process to produce the regression equation in the form of Eq.(1.2) using this example and discuss the results of the prediction accuracy evaluation.

2 Carrying Out Smoothing Splines

To construct a regression equation in the form of Eq.(1.2), we created functions ($\{x_{ik}(t)\}(1 \leq i \leq n, 1 \leq k \leq K)$) that constitute predictors of the data. All the values of $\{x_{ik}(t)\}$ for continuous t cannot be obtained in usual experiments or censuses; instead, values of $\{X_{ik}(t_{ijk})\}$ for discrete t are available. Hence, we assume that $\{X_{ik}(t_{ijk})\}$ are given for $\{t_{ijk}\}$ $(1 \leq j \leq m(i,k), t_{i1k} < t_{i2k} < \ldots < t_{i,m(i,k),k})$.

To derive $\{x_{ik}(t)\}$ using $\{X_{ik}(t_{ijk})\}$, we used the smoothing spline technique (e.g., as described in [14]). That is, we obtained $\{x_{ik}(t)\}$ that minimize E_{ik} as follows:

$$E_{ik} = \sum_{j=1}^{m(i,k)} \left(X_{ik}(t_{ijk}) - x_{ik}(t_{ijk}) \right)^2 + \lambda \int_{x_1}^{x_n} \left(\frac{d^2 x_{ik}(t)}{dt^2} \right)^2 dt, \tag{2.1}$$

where λ is a positive constant called the smoothing parameter. λ was optimized using GCV (generalized cross-validation) [15].

Next, the following equation was used to define $x_{ik}(t)$.

$$x_{ik}(t) = \sum_{p=1}^{b(i,k)} c_{ikp}\phi_{ikp}(t), \tag{2.2}$$

where $\{\phi_{ikp}(t)\}$ are cubic B-spline bases (hereafter referred to as B-spline base). $\{c_{ikp}\}$ are coefficients of B-spline bases and $b(i,k)$ is the number of bases. The functions described above are termed spline functions. If a specific positive value is given as λ in Eq.(2.1) and E_{ik} is minimized, $\{c_{ikp}\}$ are obtained, which gives $x_{ik}(t)$.

For a simple example with a setting of $i = 1$ and $k = 1$, Eq.(2.2) was transformed into

$$x_{11}(t) = \sum_{p=1}^{b(1,1)} c_{11p}\phi_{11p}(t). \tag{2.3}$$

By abbreviating the suffixes of i and k, we have

$$x(t) = \sum_{p=1}^{b(1,1)} c_p\phi_p(t). \tag{2.4}$$

Furthermore, we assumed $\{t_j\} = \{-1.5, -0.5, 1.6, 2.3, 4, 5.1, 6, 7.1\}$ and $\{X_{11}(t_j)\} = \{-3.3, -2.5, 1, 4.9, 5.9, 6.6, 6.2, 7.9\}$. $m = 7$ was set as an example. In R language, $\{t_j\}$ is represented as `t1` and $\{X_{11}(t_j)\}$ is represented as `x1`. $x(t)$ (Eq.(2.4)) given by Eq.(2.1) and GCV are obtained by the R command as:

```
xbas1  <- create.bspline.basis(c(-2 ,8), nbasis = 7)
xsb1 <- smooth.basis(t1, x1, xbas1)
xfd1   <- xsb1$fd
et1 <- seq(from = -2, to = 8, length = 100)
ex1 <- eval.fd(et1, xfd1)
```

Here, `create.bspline.basis(c(-2 ,8), nbasis = 7)` constructs seven B-spline bases in the range -2 to 8. Because the positions of knots were not specified here, equi-spaced knots were set. `et1` contains 100 equi-spaced values in this range. Then, `eval.fd()` yields values of $\hat{x}(t)$ (the function given by optimizing $x(t)$) at `et1`. The data points given by `t1` and `x1` together with those given by `ex1` are plotted in Fig.1 (left). `ex1` contains the values of the curve that were obtained by setting Eq.(2.2) and minimizing Eq.(2.1); these values correspond to `et1`. λ was optimized using GCV. It is clear from the plot, that moderate smoothing of the data points led to $\hat{x}(t)$. The above description is an outline of deriving $\hat{x}(t)$ using smoothing splines.

Furthermore, the `xfd1` obtained here is a "functional data object" (as defined on page 245 in [13]) that stores the results of smoothing the data, which consist of `t1` and `x1`. The contents of `xfd1` is shown by executing the R command:

```
plot(xfd1, ylim = c(-5.5, 9.5), xlab = expression(t[1]),
ylab = expression(x[1]), mgp = c(1.9, 1, 0))
```

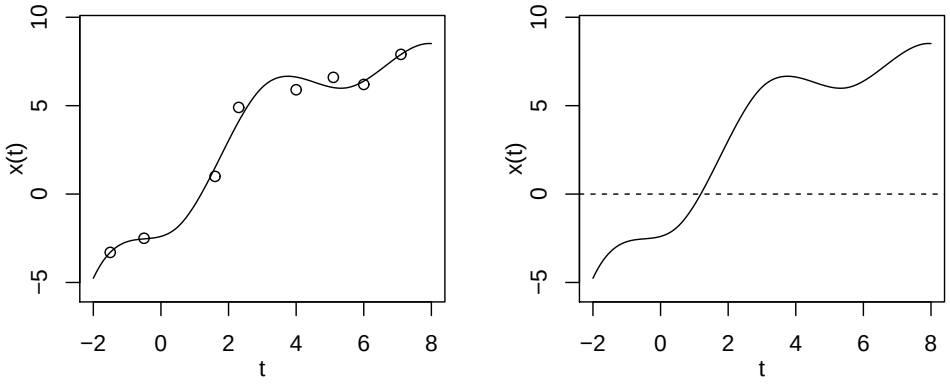

Figure 1: Optimized $x(t)$ (i.e. $\hat{x}(t)$). Data points given by $\{t_j\}$ and $\{X_{11}(t_j)\}$ (◯). $\hat{x}(t)$ was given by Eq.(2.1) and GCV (solid line) (left). Optimized $x(t)$ (i.e. $\hat{x}(t)$) given by another R command. $\hat{x}(t)$ was given by `xfd1`. Equi-spaced knots were set (right).

This R command was used to draw Fig.1 (right). The two curves (Fig.1) are identical.

To articulate clearly the functions that constructed the ($\hat{x}(t)$) curves shown in Fig.1, second order derivatives of the curves were calculated as:

```
dex1 <- eval.fd(et1, xfd1, Lfdobj = 2)
```

The `dex1` value yielded by this R command are second order derivatives of $\hat{x}(t)$ at `et1`. `Lfdobj = 2` specifies that second order derivatives are to be estimated. `dex1` is illustrated in the left panel in Fig.2, which indicates that $\hat{x}(t)$ consists of four regions given by five points ($\{-2, 0.5, 3, 5.5, 8\}$) and that the functions in the respective regions are cubic. Moreover, $\hat{x}(t)$ and its second order derivative are continuous at boundary points of the four regions. The points at $\{-2, 0.5, 3, 5.5, 8\}$ are called knots. The knots at both ends are termed quadruple knots because, to construct B-spline bases, four knots are overlapped at these points . Here, the number of bases is set at seven; hence, the number of knots is five if each quadruple knot is considered one knot. (The number of bases subtracted by 2 equals the number of knots. This rule holds for any number of bases.)

The following R command was used to draw the seven B-spline bases used here.

```
plot(xbas1, col = 1, xlab = expression(t[1]), ylab = expression(x[1]),
 mgp=c(1.9, 1, 0))
```

The behaviors of the seven B-spline bases is shown in right panel in Fig.2.

3 Producing a Functional Data Multiple Regression Equation

Using $\hat{x}(t)$, which was obtained as $\{x_{ik}(t)\}$ in Eq.(1.2), as described in Section 2, a functional data multiple regression equation in the form of Eq.(1.2) is produced. Here, simple simulation data are used to illustrate the procedures.

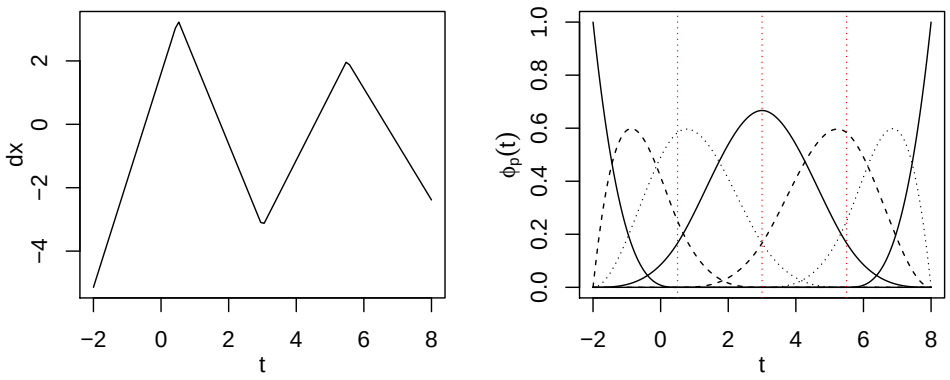

Figure 2: Second order derivatives of $\hat{x}(t)$ from Fig.1. Seven B-spline bases are given by five knots placed at $\{-2, 0.5, 3, 5.5, 8\}$ (left). Seven B-spline bases given by 5 knots placed at $\{-2, 0.5, 3, 5.5, 8\}$. These B-spline bases are used for producing the curve in the left figure. The knots at both ends are quadruple knots (right).

Set $n = 50$, $K = 2$, $m(i, 1) = m(i, 2) = 30$ $(1 \leq i \leq 50)$, and $\{t_{ijk}\} = j$ $(1 \leq i \leq 50, 1 \leq k \leq 2)$; i.e., the number of data (n) is set at 50, the number of predictors of the functional data multiple regression equation (K) is set at 2, and the numbers of data that yield respective predictors in the functional data multiple regression equation $(m(i, 1)$ and $m(i, 2))$ are both set at 30 for all values of i. All the values of $\{X_{ik}(t_{ijk})\}$ $(1 \leq i \leq 50, 1 \leq j \leq 30, 1 \leq k \leq 2)$ are realizations of the uniform random numbers in which the minimum value is 0 and the maximum value is 1. The values of $\{y_i\}(1 \leq i \leq 50)$ are given by the equation:

$$y_i = 3\sum_{j=1}^{30} X_{ij1}\mathrm{sin}(1.5\pi j/30) + 5\sum_{j=1}^{30} X_{ij2}\mathrm{cos}(2.5\pi j/30) + \epsilon_i. \tag{3.1}$$

The $\{\epsilon_i\}$ values are realizations of $N(5, 2^2)$ (a normal distribution with mean 0 and variance 2^2). When $X_{i1}(t_{ij1})$ is depicted as `xx1[jj, ii]` and $X_{i2}(t_{ij2})$ is depicted as `xx2[jj, ii]`, then `xx1[jj, ii]` and `xx2[jj, ii]` are obtained using the R command:

```
library(fda)
nd <- 50
nx <- 30
xx1 <- matrix(rep(0, length = nd * nx), ncol = nd)
xx2 <- matrix(rep(0, length = nd * nx), ncol = nd)
yy <-NULLs
for(ii in 1:nd){
  xx1[, ii] <- runif(nx, min = 0, max = 1)
  xx2[, ii] <- runif(nx, min = 0, max = 1)
  ss <- 0
  for(jj in 1:nx){
    ss <- ss + xx1[jj, ii] * 3* sin(1.5 * pi * jj / nx) +
     xx2[jj, ii] * 5 * cos(2.5 * pi * jj / nx)
```

```
    }
    yy[ii] <-  ss + 5 + rnorm(1, mean = 0, sd = 2.0)
  }
```

Here, `library(fda)` indicates that package "fda" was used. `nd` is the number of data (n) and `nx` represents both $m(i,1)$ and $m(i,2)$.

Next, B-spline bases are constructed to be stored in `xb1` and `xb2` using the following R command:

```
xb1  <- create.bspline.basis(c(0.5, nx + 0.5), nbasis = 20)
xb2  <- create.bspline.basis(c(0.5, nx + 0.5), nbasis = 20)
```

In this R command, the range where B-spline bases exist is set to be from 0.5 to 30.5; i.e., the positions of the quadruple knots are $t = 0.5$ and $t = 30.5$. Because `nbasis = 20` is set and the positions of knots (except the quadruple knots) are not specified, 20 B-spline bases are located at equi-spaced positions in this range.

Then, `xs1` and `xs2` are constructed using `xx1[jj, ii]` and `xx2[jj, ii]`, where `xs1` and `xs2` stand for $\{x_{i1}(t)\}(1 \leq i \leq 50)$ and $\{x_{i2}(t)\}(1 \leq i \leq 50)$, respectively. The following R command is used for this purpose.

```
xs1 <- smooth.basis(tt, xx1, xb1)
xs2 <- smooth.basis(tt, xx2, xb2)
```

To create `xs1` and `xs2`, smoothing spline is carried out using B-spline bases specified by `xb1` and `xb2`, respectively. The value of smoothing parameter (λ in Eq.(2.1)) is not assigned in this R command; hence, the value of the smoothing parameter is optimized by GCV. Furthermore, the values of the smoothing parameters of $\{x_{i1}(t)\}(1 \leq i \leq 50)$ for all of $\{x_{i1}(t)\}(1 \leq i \leq 50)$ are identical.

The `fd`-component is extracted from `xs1` and `xs2` as follows:

```
xf1   <- xs1$fd
xf2   <- xs2$fd
```

where `fd`-component is a functional data object that contains basis functions, which in this example are B-spline bases (i.e., $\{\phi_{ikp}(t)\}$ in Eq.(2.2), and the coefficients of basis functions, which in this example are $\{c_{ikp}\}$ in Eq.(2.2).[MB24]

Next, to produce a functional data multiple regression equation using `xf1` and `xf2`, these two objects were combined as follows:

```
xl      <- vector("list",2)
xl[[1]] <- rep(1,nd)
xl[[2]] <- xf1
xl[[3]] <- xf2
```

Here, `xl[[1]]` indicates the presence of a constant term (i.e., α_0 in Eq.(1.2)) in this functional data multiple regression equation. `xl[[2]]` indicates the presence of the first term of predictors ($\int_{t_1^{min}}^{t_1^{max}} x_{i1}(t)\beta_1(t)dt$ in Eq.(1.2)) and `xl[[3]]` indicates the presence of the second term of predictors ($\int_{t_2^{min}}^{t_2^{max}} x_{i2}(t)\beta_2(t)dt$ in Eq.(1.2)).

Next, B-spline bases were created to construct $\{\beta_k(t)\}$ (Eq.(1.2)) using the following R command:

```
bb1 <- create.bspline.basis(c(0.5, nx + 0.5), nbasis = 20)
bb2 <- create.bspline.basis(c(0.5, nx + 0.5), nbasis = 20)
```

Here, bb1 gives B-spline bases for $\{\beta_1(t)\}$ and bb2 gives B-spline bases for $\{\beta_2(t)\}$; the specifications of xb1 and xb2 are identical.

To prepare for the creation of a functional data multiple regression equation, bp1 and bp2 are obtained using bp1 and bp2 using the following R command:

```
bp1  <- fdPar(bb1, Lfdobj = 2)
bp2  <- fdPar(bb2, Lfdobj = 2)
```

where Lfdobj = 2 specifies that the equation containing second order derivatives of $\{\beta_k(t)\}$ (Eq.(1.2) is used for smoothing $\{\beta_k(t)\}$ when $\{\beta_k(t)\}$ are produced. If lambda = is contained in the arguments of fdPar(), the smoothing parameter (λ in Eq.(2.1)) is specified. However, in this example, the smoothing parameter was not assigned because it is specified in a later process.

Next, bp1 and bp2 were combined into bla as follows:

```
bla <- vector("list",2)
conbasis   <- create.constant.basis()
bla[[1]] <- conbasis
bla[[2]] <- bp1
bla[[3]] <- bp2
```

This process is similar to the one used for xl.

Then, the smoothing parameter used to produce $\{\beta_k(t)\}$ (Eq.(1.2)) was optimized by GCV as follows:

```
lams <- 10^(seq(from = 0, to = 3, by = 0.2))
gcvs <- rep(0, length(lams))
for(kk in 1:length(lams)){
  blb <- bla
  par2 <- blb[[2]]
  par2$lambda <- lams[kk]
  blb[[2]] <- par2
  par3 <- blb[[3]]
  par3$lambda <- lams[kk]
  blb[[3]] <- par3
  reg1 <- fRegress(yy, xl, blb)
  gcvs[kk] <- reg1$gcv
}
wh1 <- which(gcvs == min(gcvs))
lbest <- lams[wh1]
```

The values of the smoothing parameters were set first as lams. In this example, they are $\{10^0, 10^{0.2}, 10^{0.4}, \cdots, 10^3\}$. The values of GCV given by the smoothing parameters are stored in gcvs. The $lambda stored in [[2]]-component of blb is the smoothing parameter for deriving $\beta_1(t)$; blb is identical to bla. The $lambda stored in [[3]]-component is the smoothing parameter for deriving $\beta_2(t)$. In this example, the fRegress() R command, which performs regression based on functional data analysis, was carried out by altering the values of the smoothing parameters. $gcv-component stored in reg1, which is the output of fRegress() gives GCV. Hence, lbest represents the best λ.

The relationship between the smoothing parameters (λ) and GCV was obtained by:

```
plot(lams, gcvs, type = 'n', log = 'x', mgp=c(1.9, 1, 0))
lines(lams, gcvs)
points(lams, gcvs)
```

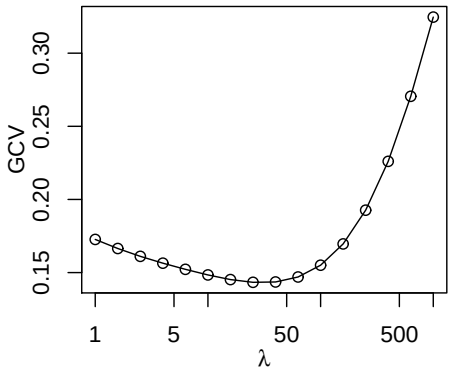

Figure 3: Relationship between the smoothing parameter (λ) and GCV.

The following R command was used to draw Fig.3.

```
blb <- bla
par2 <- blb[[2]]
par2$lambda <- lbest
blb[[2]] <- par2
par3 <- blb[[3]]
par3$lambda <- lbest
blb[[3]] <- par3
reg2 <- fRegress(yy, xl, blb)
bl2   <- reg2$betaestlist
bp1 <- bl2[[2]]
bf1 <- bp1$fd
bp2 <- bl2[[3]]
bf2 <- bp2$fd
```

The derived $\beta_1(t)$ and $\beta_2(t)$ functions are stored in `reg2`. The following R command was used to draw Fig.4,

```
et1 <- seq(from = 0.5, to = nx + 0.5, by = 0.5)
eb1 <- eval.fd(et1, bf1)
plot(et1, eb1, xlab = "t", ylab = expression(beta[1](t)),
 type="n", mgp=c(1.9, 1, 0))
lines(et1, eb1)
et2 <- seq(from = 0.5, to = nx + 0.5, by = 0.5)
eb2 <- eval.fd(et2, bf2)
plot(et2, eb2, xlab = "t", ylab = expression(beta[2](t)),
 type="n", mgp=c(1.9, 1, 0))
lines(et2, eb2)
```

Here, `eval.fd()` calculates the values of a function at `et1` or `et2` using the functional data object, which in this example, are `bf1` or `bf2`, respectively (Fig.4(left)(right)).

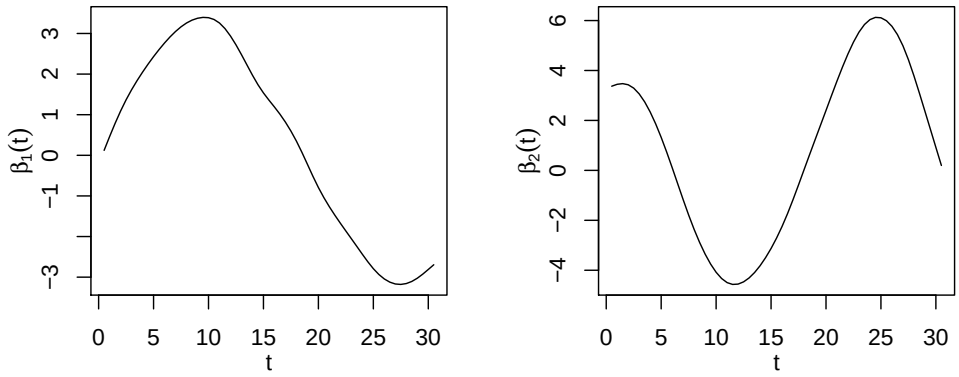

Figure 4: $\hat{\beta}_1(t)$ (left) and $\hat{\beta}_2(t)$ (right) given by the functional data objects `bf1` or `bf2`, respectively.

The estimated values of the target variable (`eyy`) that corresponds to the predictors of data ($\{y_i\}$) were extracted by:

```
eyy <-  reg2$yhatfdobj
```

Moreover, the constant term, α_0, in Eq.(1.2) was derived by:

```
bp0 <- bl2[[1]]
bf0 <- bp0$fd
alpha0 <- bf0$coef
```

where `alpha0` is the estimate of α_0.

The procedures described here were used to successfully construct a functional data multiple regression equation and output the contents of the result. When arbitrary $\{\hat{x}_{ik}(t)\}$ was used (Eq.(1.2)), $\{\hat{\beta}_k(t)\hat{x}_{ik}(t)\}$ are to be integrated to calculate the value of the target variable. In the following section, we use SNP data and the composite Simpson's rule (see page 257 of [16] for details) to carry out the numerical integration.

4 Prediction of Breeding Values

The application of functional data analysis for predicting breeding values using SNP data is described. The focus is on prediction of breeding values for breeding crop plants and livestock.

Breeding methods consists of (1) selection of individuals and varieties on the basis of desirable genetic traits such as yields or productivities, (2) repeated hybridizations over several generations, and (3) foundation of individuals and breed varieties with the desirable genetic traits. Quantitative traits such as the amount of yield or growth, which are represented as continuous values, are not directly observed; instead, phenotypic values associated with the trait are observed. The phenotypic value of a trait consists of the genotypic value and non-genetic factors such as environment. Therefore,

the influence of non-genetic noise should be reduced to accurately estimate breeding values. In the past, breeding values were predicted using phenotypic values of many offspring or relatives. For example, breeding values of self-pollinating plants can be predicted on the basis of the observations of and/or values of genetically identical plants using the same inbred lines; however, the prediction of breeding values in this way is laborious, time consuming, and expensive. For example, observation of phenotypic values of fruit traits of fruit plants needs more than several years. Because determining phenotypic values for a desirable trait contributes most to the cost of breeding using traditional methods, genomic information is now being used as a more efficient way of estimating breeding values.

It has been shown that quantitative traits are influenced by a large number of loci scattered along a genome with each loci having a small effect. The combination of these small effects and the non-genetic effects determine the phenotypic value of a quantitative trait. Generally, in conventional breeding methods using genomic information, the influence of one or two genes is focused on and a small number of DNA markers located in the genome near the genes are used for selection. Such methods, however, do not realize efficient breeding if a large number of genetic loci are involved in a trait. Hence, methods for more accurate and efficient predictions of breeding values for quantitative traits have received much attention recently. These methods attempt to consider the influences of a large number of genetic loci on quantitative traits using detailed information about the genes. Genome-wide information has been accumulating at a very steep rate for many years. A new genomic selection method [2] has been developed that selects desirable individuals and varieties based on breeding values predicted using a large number of SNP markers arranged densely in a whole genome. This method does not require phenotypic values of individuals and varieties as input. The feasibility and usability of this method has been investigated [17].

To predict breeding values using SNP information, prediction models that use large data sets of SNP genotypes from individuals and varieties along with the corresponding phenotypic values need to be developed. These data sets can be used as training data to estimate parameters for the proposed models. If the SNP genotypes in the training data are set as X, the phenotypic values are set as Y, and the prediction error is set as e, then a model can be represented as $Y = f(X) + e$, where $f(X)$ is a term that resulted from the regression of X on Y. If X contained all the information on a whole genome, then $f(X)$ could be regarded as the breeding values, which are hereditarily determined. Furthermore, using X^*, which are the SNP genotype data for the individuals or breed varieties to be selected, the breeding values of these individuals or varieties is given as $f(X^*)$. Because the genomic information for an individual can be collected before birth or germination, the time-span needed for breeding will be shortened dramatically by selecting and growing new individuals, breed varieties, or cultivars using the predicted breeding values without phenotypic observations. Clearly, the success of such a method will depend heavily on the accuracy of $f(X^*)$.

Actual breeding values are not observable; therefore, to test the accuracy of a proposed prediction method, simulation data instead of real data can be employed. The simulation data were generated by setting genetic loci that affect the traits and the arrangement and number of SNPs on a genome, and assuming the effects of the non-genetic environment. In this way, phenotypic values and SNP genotype data were generated as the training data. Independent validation data were simulated using real breeding values and the corresponding SNP genotype data. These data are treated as real breeding values and compared with the predicted breeding values to determine the accuracy of the proposed prediction method.

5 Functional Data Multiple Regression Equation Given by Simulated SNP and Traits Data

The simulation data used here is from [18]. These data consist of training data that were generated assuming $1,000$ outbred diploid individuals and validation data that consisted of $1,000$ SNP data and

associated trait data.

The analysis of the first data set among the 20 data sets ([18]) is described here. The predictors are the SNP genotype data each of which takes one value among $\{0, 1, 2\}$. The number of predictors is $10,100$, comprising 10 sections of $1,010$ values. In each of the 10 sections, it is assumed that the effects of neighboring predictors are similar; however, this similarity does not cross the of section boundaries. The target variable of this data are traits. The number of trait data is $2,000$. Hence, to produce a functional data multiple regression equation, the following data is considered: $n = 2,000$, $K = 10$, $m(i,1) = m(i,2) = \ldots = m(i,10) = 1,010$ $(1 \leq i \leq n)$A$\{t_{ijk}\} = j$ $(1 \leq i \leq n, 1 \leq k \leq 10)$. Furthermore, we set $t_1^{min} = t_2^{min} = \ldots, = t_{10}^{min} = 0.5$, $t_1^{max} = t_2^{max} = \ldots, = t_{10}^{max} = 1,000.5$. Each $\{x_{ik}(t)\}$ and $\{\beta_k(t)\}$ is represented by 100 B-spline bases with knots located at equi-spaced points. One half of these $2,000$ data (i.e., $1,000$ data) are used to produce a functional data multiple regression equation using the R command described in the section 3. The smoothing parameter (λ) is optimized by GCV as illustrated in Fig.5, which shows that $\lambda = 10^{6.2}$ $(= 1,584,893)$ is optimal.

This optimal value of λ is used to construct a functional data multiple regression equation illustrated in Figs.6 and 7. The estimate of α_0 (Eq.(1.2)) in this regression equation is $\hat{\alpha}_0 = -4.349619$.

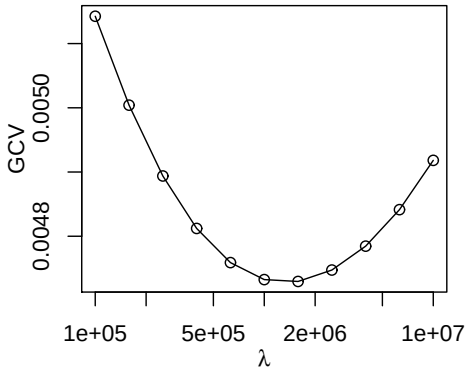

Figure 5: Relationship between smoothing parameter (λ) and GCV. The optimal λ is relationship is used to construct a functional data multiple regression equation.

When the target variables of the data are represented as $\{y_i\}$ $(1 \leq i \leq 1,000)$ and the estimates corresponding to these values are represented as $\{\hat{y}_i\}$ $(1 \leq i \leq 1,000)$, the relationship between $\{y_i\}$ and $\{\hat{y}_i\}$ is shown in Fig. 8(left). Using the resulting functional data multiple regression equation, the values of the target variables are estimated using the other half of the data ($\{y_i^{pre}\}$ $(1 \leq i \leq 1,000)$) (i.e., another $1,000$ data). The estimated values are termed $\{\hat{y}_i^{pre}\}$ $(1 \leq i \leq 1,000)$ and the relationship between $\{y_i^{pre}\}$ and $\{\hat{y}_i^{pre}\}$ is illustrated in Fig.8(right).

The discrepancy between $\{y_i^{pre}\}$ and $\{\hat{y}_i^{pre}\}$ is calculated as

$$E^{pre} = \frac{\sum_{i=1}^{1,000}(y_i^{pre} - \hat{y}_i^{pre})^2}{1,000}. \tag{5.1}$$

We obtained an estimate of $E^{pre} = 4.383985$. Here, it is important that the magnitude among the estimates is similar to the magnitude among the data. To check this relation, the $Corr^{pre}$ (Pearson's

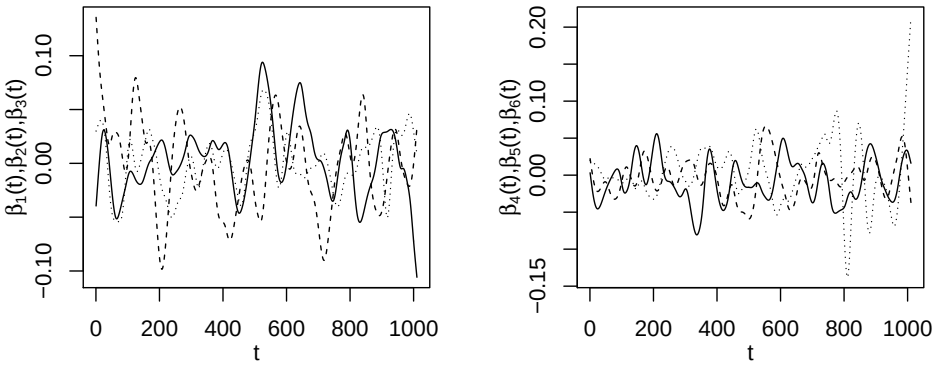

Figure 6: Regression equation represented as $\hat{\beta}_1(t)$ (solid line), $\hat{\beta}_2(t)$ (broken line), $\hat{\beta}_3(t)$ (dotted line) (left) and $\hat{\beta}_4(t)$ (solid line), $\hat{\beta}_5(t)$ (broken line), $\hat{\beta}_6(t)$ (dotted line) (right).

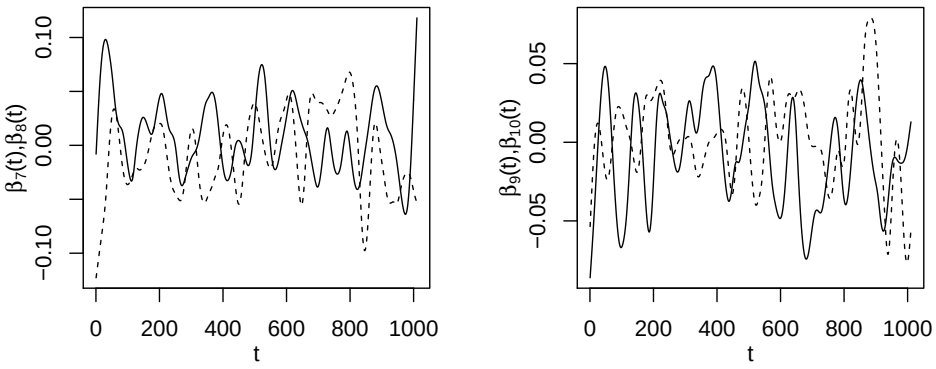

Figure 7: Regression equation represented as $\hat{\beta}_7(t)$ (solid line), $\hat{\beta}_8(t)$ (broken line) (left) and $\hat{\beta}_9(t)$ (solid line), $\hat{\beta}_{10}(t)$ (broken line) (right).

product-moment correlation coefficient) was calculated as:

$$Corr^{pre} = \frac{\sum_{i=1}^{1,000}(y_i^{pre} - \bar{y}^{pre})(\hat{y}_i^{pre} - \bar{\hat{y}}^{pre})}{\sqrt{\sum_{i=1}^{1,000}(y_i^{pre} - \bar{y}^{pre})^2}\sqrt{\sum_{i=1}^{1,000}(\hat{y}_i^{pre} - \bar{\hat{y}}^{pre})^2}}, \tag{5.2}$$

where $\bar{y}^{pre}$ and $\bar{\hat{y}}^{pre}$ were defined as

$$\bar{y}^{pre} = \frac{\sum_{i=1}^{1,000} y_i^{pre}}{1,000}, \qquad \bar{\hat{y}}^{pre} = \frac{\sum_{i=1}^{1,000} \hat{y}_i^{pre}}{1,000}. \tag{5.3}$$

The $Corr^{pre}$ was calculated as equal to 0.6590756.

A multiple regression equation is produced for comparison. By averaging the $1,010$ predictor data in each section, 10 predictor data were obtained and a multiple linear regression equation was constructed using these data. That is, a multiple linear regression equation with 10 predictor was obtained. When the one half of this $2,000$ data (i.e., $1,000$ data) was used to produce a multiple linear regression equation, the relationship between $\{y_i\}$ and $\{\hat{y}_i\}$ is illustrated in Fig.9(left). Using the resulting multiple linear regression equation, the values of the target variables were estimated using the other half of the data ($\{y_i^{pre}\}$ $(1 \leq i \leq 1,000)$) (i.e., the remaining $1,000$ data). The estimated values are termed $\{\hat{y}_i^{pre}\}$ $(1 \leq i \leq 1,000)$. The relationship between $\{y_i^{pre}\}$ and $\{\hat{y}_i^{pre}\}$ is illustrated in Fig.9(right). Moreover, $E^{pre} = 7.305007$ and $Corr^{pre} = 0.1644215$. By comparing the left and right graphs in Fig.8, we found that the functional data multiple regression equation performed better than the multiple linear regression equation.

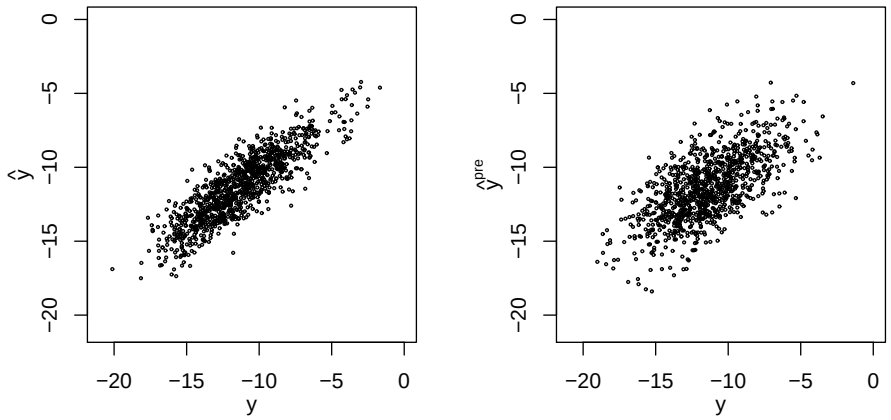

Figure 8: Relationship between $\{y_i\}$ and $\{\hat{y}_i\}$ (left). Relationship between $\{y_i^{pre}\}$ and $\{\hat{y}_i^{pre}\}$ (right).

Using the 19 data sets and the same method as that used for the first $2,000$ data, similar functional data multiple regression equations were derived and E^{pre} (Eq.(5.1)) and $Corr^{pre}$ (Eq.(5.2)) was calculated. The results are shown in Fig.10. The first of the data sets is the data set that was used in the previous example. The results indicate that functional data multiple regression equations predict breeding values of traits with considerable accuracy.

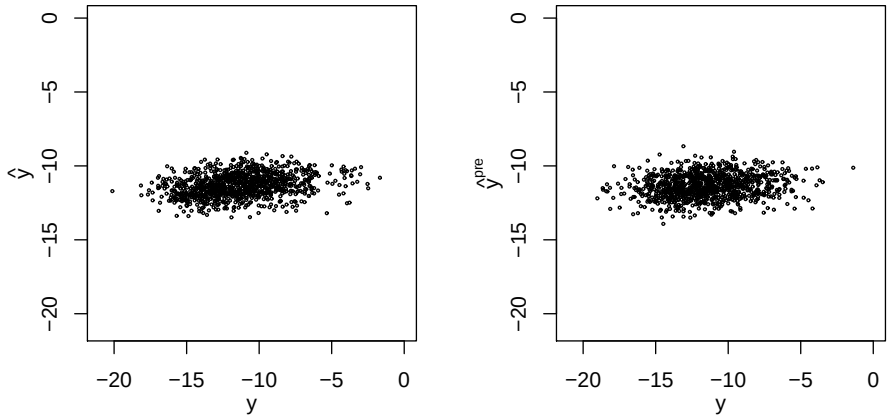

Figure 9: Relationship between $\{y_i\}$ and $\{\hat{y}_i\}$ when a multiple linear regression equation is produced (left). Relationship between $\{y_i^{pre}\}$ and $\{\hat{y}_i^{pre}\}$ (right).

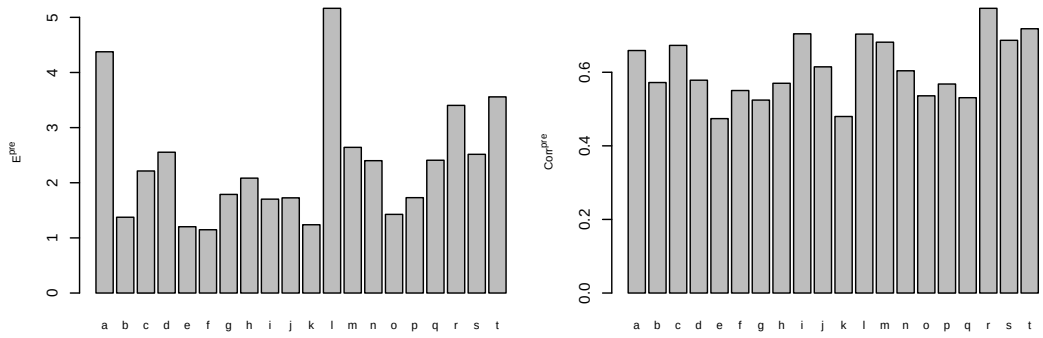

Figure 10: E^{pre} given by 20 data sets (a o t) (left). $Corr^{pre}$ given by 20 data sets (a to t) (right).

6 Conclusions

In this study, we show that the simple functional data analysis R package “fda” (version 2.4.0) can be used to produce functional data multiple regression equations. When the number of predictors is very large and more than the amount of available data, as can be the case when predicting breeding values using SNP genotype data, functional data multiple regression allows the automatic derivation of beneficial regression equations; optimization of the smoothing parameters is also automatic. This is a major advantage of this technique when analyzing data of this kind. Even if the prediction errors are kept small by selecting predictors or by adjusting prior distributions by trial and error, it is still uncertain whether or not the obtained prediction errors represent the essential prediction errors.

Furthermore, in conventional data analysis, sometimes several data are averaged for use in regression analyses, or an effect is assumed to be constant even when it may be time-dependent. Such treatments are adopted so that the regression equation can be reduced to multiple linear regression. Although many predictors may be useful for predicting the values of a target variable, in many instances, a small number of predictors are selected because efficient methods for treating large numbers of predictors are not available. We expect that functional data multiple regression and higher-level regression methods using package “fda” will be used in future prediction models, or other prediction models that can use data in which the number of predictors is much more than the amount of data will be developed before long.

Acknowledgements

The author is very grateful to the referees for carefully reading the paper and for their comments and suggestions which have improved the paper.

Competing Interests

The author declares that no competing interests exist.

References

[1] Pavlopoulos GA, Oulas A, Iacucci E, Sifrim A, Moreau Y, Schneider R, Aerts J, Iliopoulos I. Unraveling genomic variation from next generation sequencing data. BioData Mining. 2013;6:13.

[2] Meuwissen TH, Hayes BJ, Goddard ME. Prediction of total genetic value using genome-wide dense marker maps. Genetics. 2001;157:1819-1829.

[3] Gianola D, van Kaam J. B. C. H. M. Reproducing kernel Hilbert spaces methods for genomic assisted prediction of quantitative traits. Genetics. 2008;78:2289-2303.

[4] Usai MG, Goddard ME, Hayes BJ. LASSO with cross-validation for genomic selection. Genetics Research. 2009;91:427-436.

[5] Neves HHR, Carvalheiro R, Queiroz SA. A comparison of statistical methods for genomic selection in a mice population. BMC Genetics. 2012;13:100.

[6] Ogutu JO, Schulz-Streeck T, Piepho HP. Genomic selection using regularized linear regression models: ridge regression, lasso, elastic net and their extensions. BMC Proceedings. 2012;6(Suppl 2):S10.

[7] Resende Jr. MFR, Muñoz P, Resende MDV, Garrick DJ, Fernando RL, Davis JM, Jokela EJ, Martin TA, Peter GF, Kirst M. Accuracy of Genomic Selection Methods in a Standard Dataset of Loblolly Pine (Pinus taeda L.). Genetics. 2012;190:1503-1510.

[8] Sun X, Ma P, Mumm RH. Nonparametric Method for Genomics-Based Prediction of Performance of Quantitative Traits Involving Epistasis in Plant Breeding. PLoS ONE. 2012;7:e50604.

[9] Ramsay JO, Silverman BW. Applied Functional Data Analysis: Methods and Case Studies;2002. New York: Springer.

[10] Ramsay JO, Silverman BW. Functional Data Analysis, 2nd edition;2005. New York: Springer.

[11] Takezawa K. Introduction to Nonparametric Regression;2006. Hoboken: Wiley.

[12] Ramsay JO, Hooker G, Graves S. Functional Data Analysis with R and MATLAB (Use R!);2009. New York: Springer.

[13] Ramsay JO, Wickham H, Graves S, Hooker G. Functional Data Analysis. Version 2.4.0 Date 2013-05;2013.
Available: http://cran.r-project.org/web/packages/fda/fda.pdf

[14] Green PJ, Silverman BW. Nonparametric Regression and Generalized Linear Models: A roughness penalty approach;1993. London: Chapman and Hall/CRC.

[15] Craven P, Wahba G. Smoothing noisy data with spline functions. Numerische Mathematik. 1979;31:377-403.

[16] Atkinson K. An Introduction to Numerical Analysis,second edition;1989. New Jersey: Wiley.

[17] Lorenz AJ, Chao S, Asoro FG, Heffner EL, Hayashi T, Iwata H, Smith KP, Sorrells ME, Jannink J. Genomic selection in plant breeding: knowledge and prospects. Advances in Agronomy. 2011;110:77-123.

[18] Hayashi T, Iwata H. EM algorithm for Bayesian estimation of genomic breeding values. Genetics. 2010;11:3.

Peer-review history:
The peer review history for this paper can be accessed here (Please copy paste the total link in your browser address bar)
www.sciencedomain.org/review-history.php?iid=936&id=6&aid=8218