S+ Functional Data Analysis

Douglas B. Clarkson
Chris Fraley
Charles C. Gu
James O. Ramsey

S+ Functional Data Analysis

User's Manual for Windows®

With 79 Illustrations

Springer

Douglas B. Clarkson
Insightful Corporation
1700 Westlake Ave. N.
Seattle, WA 98109

Charles C. Gu
Insightful Corporation
1700 Westlake Ave. N.
Seattle, WA 98109

Chris Fraley
Insightful Corporation
1700 Westlake Ave. N.
Seattle, WA 98109

James O. Ramsey
McGill University
1025 Dr. Penfield Ave
Montreal, Quebec H3A 1B1

With 79 illustrations.

Library of Congress Control Number: 2005924801

ISBN-10: 0-387-24969-9 Printed on acid-free paper.
ISBN-13: 978-0387-24969-9

Printed in the United States of America. (MVA)

9 8 7 6 5 4 3 2 1 11371380

springeronline.com

ACKNOWLEDGMENTS

The software for the Functional Data Analysis module was originally written by Jim Ramsay, Department of Psychology, McGill University, and Bernard Silverman, Department of Mathematics, University of Bristol. We have contributed enhancements and extensions, and attempted to reflect their zeal for the analysis of functional data. We have benefited from contributions by James Schimert, and comments by Tim Hesterberg at Insightful Corporation. Our efforts were funded by NIH SBIR grants 1R43CA86539-01 and 2R44CA86539-02 entitled: *An S-Plus Functional Data Analysis Module.*

CONTENTS

PREFACE

The book is intended as a guide to the functional data analysis software in the S+FDA library. It gives a general overview, and treats each topic through illustrative examples. The code for the examples can be found in the script files provided with the software, which also include additional examples. Users can learn to use the S+FDA library by executing the example scripts while reading. Details on the functions and their arguments, as well as further examples, can be found in the associated help files.

INTRODUCTION

1

Functional data arise in many fields of research. Measurements are often best thought of as functions, even in cases where the data is gathered at a relatively small number of points. Examples include weather changes, stock prices, bone shapes, growth rates, health status indicators, and tumor size.

For time-dependent data, observations may be viewed as realizations of a smooth function $y(t)$ of time that have been measured (with error) at specific time points t_j, but which could have been measured at any time. Spatial functional data is also common, e.g., the length of a bone along an axis, the concentration of a drug in a tissue as a function of depth, yearly mean temperature as a function of location.

Historically, functional data have been analyzed using multivariate or time-series methods at discrete measurement points. Analyzing functional data instead as functions has several advantages:

- Functions, unlike raw data, can be evaluated at any “time” point. This is important because it allows the use of statistical methods requiring evenly-spaced measurements and allows extrapolation for use in predictions or treatment decisions.
- Functional methods (e.g., functional principal components, functional canonical correlation) apply even when the data have been gathered at irregular intervals, or at different times on different subjects, when multivariate analogues of these methods are either inappropriate or unavailable.
- Derivatives and integrals of functions may provide important information about the underlying process. For example, knowledge of the direction and rate of change of a patient’s temperature may be more important than knowledge of the patient’s current temperature.

Functional methods can also be used when the parameters to be estimated are functions. Ramsay and Silverman (1997) use smoothing spline methods for density estimation, and to estimate the link function in generalized linear models. Another example is regression splines for fitting time-dependent hazard regression models (Kooperberg and Clarkson, 1997).

S+FDA integrates functional data analysis methods into S-PLUS. It includes a complete commercial implementation of the exploratory methods of Ramsay and Silverman (1997, 2002), featuring:

- methods for transforming observed data to a smoothed functional form,
- predicting a functional or nonfunctional variable $y(t)$ as a function of one or more functional or nonfunctional variables,
- finding and rotating the functional "principal components" of a functional variable,
- finding the canonical correlations between two functional variables, and
- performing a "principal differential analysis".

S+FDA also incorporates more recent innovations and extensions, such as allowing the use of functions with arbitrary bases, and providing methods for functional generalized linear models and functional cluster analysis.

Installation

To install the software:

- Go to the website: `http://www.insightful.com/downloads/libraries/default.asp`
- Follow the on-screen Setup instructions; default settings are recommended.

Object-oriented Programming

S+FDA makes use of the object-oriented capabilities of the S-PLUS language. In object-oriented programming, constructor functions create structured data "objects" that are assigned a class (which typically has the same name as the constructor). The object-oriented paradigm allows users to apply generic functions (such as `plot`) to these classed objects, the details of which are handled transparently through class-specific functions or "methods". This simplifies programming by avoiding the need to explicitly invoke different functions or to have additional function arguments when generic operations are applied to objects of different structures.

INTRODUCTORY TUTORIAL (HEIGHT DATA)

We illustrate some exploratory functional data analysis methods using the Berkeley height data (Tuddenham and Snyder, 1954). The corresponding data frame, `heightData`, is included in the S+FDA library. This data contains the heights of 54 female (columns 2 to 55) and 39 male (columns 56 to 94) children observed at 31 times from age 1 to age 18. The times of measurement are included as the variable `age` (column 1). We first inspect the data graphically by plotting the height curves as follows:

```
#Set up the plot and label
> plot(heightData$age, heightData[,2], type="n",
       ylim=range(unlist(heightData[,2:55])),
       xlab="Age (years)", ylab="Height (cm)",
       main="Female Height Data")
#draw the height curves
> matlines(heightData$age,as.matrix(heightData[,2:55]))
```

The result is shown in Figure 1.1.

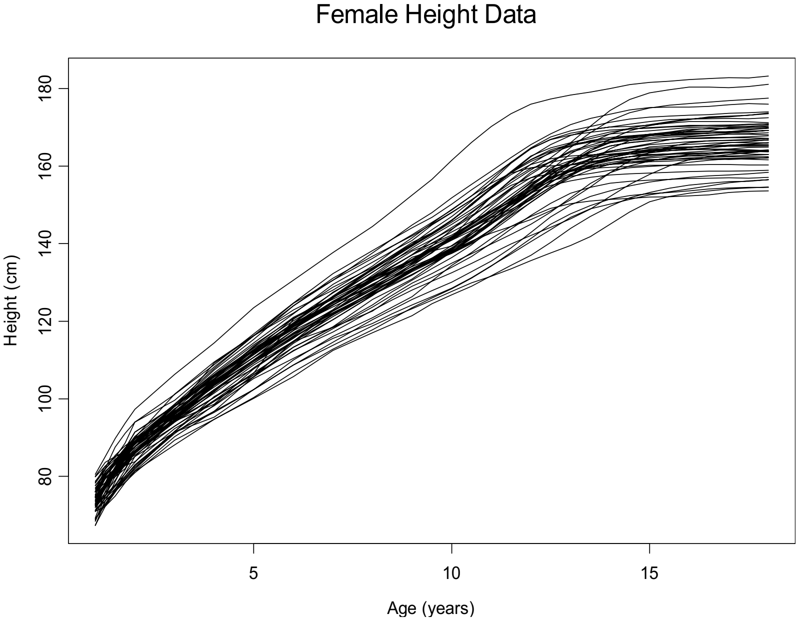

Figure 1.1: *Female height data.*

Although the data appear as smooth curves, only 31 discrete values of height were measured. The curves are produced by connecting these discrete points with straight lines.

As a functional data analysis application, we fit a function to each height curve using linear least squares. The function is represented as a linear combination of basis functions $b_j(t)$ and coefficients β_j that vary from one height function to the next:

$$f(x) = \sum_{j=1}^{n_b} \beta_j b_j(t)$$

There are a variety of choices for the basis functions, e.g., B-splines, Fourier series, and exponential series. Once the basis is chosen, the coefficients are estimated based on the observed data. In Figure 1.1, a *polygonal basis* of connected line segments is used to draw the curves.

Although the functional representation almost always differs from the data at the points of observation, these differences are assumed to be small in the sense that the coefficients β_j capture the information contained in the discretized curve. In most analyses, the raw data is ignored once the β_j have been estimated because it is simpler to work with the functional form. The assumption is that the within-subject variance in the β_j estimates is small compared to the between-subject variance.

Warning

When the number of observations for estimating the β_j is small to moderate or when the within-subject variance of the β_j estimates is large, a mixed-effects model may be preferred so that information may be combined across subjects.

Selecting the Basis Functions

To perform a functional data analysis, we must first choose an appropriate set of basis functions. In the example above, 16 B-spline basis functions of order 6 were used. Since the order of a polynomial basis is the degree plus one, this basis consists of 16 piecewise polynomial splines of degree 5. By default, the *interior* knots for the 16 basis functions are equally spaced over the range of the independent variable (the two *exterior* knots are placed at the endpoints of the function domain). Since height is being viewed as a function of age, the appropriate domain for the basis functions is the age span of the data. The following forms an object of class "bsplineBasis" for the height data:

```
> heightBasis <- bsplineBasis(fDomain
                 =range(heightData$age), nbasis=16,norder=6)
```

The basis functions, displayed in Figure 1.2, are equally spaced over the domain:

```
> plot(heightBasis, main="B-spline Basis Functions")
```

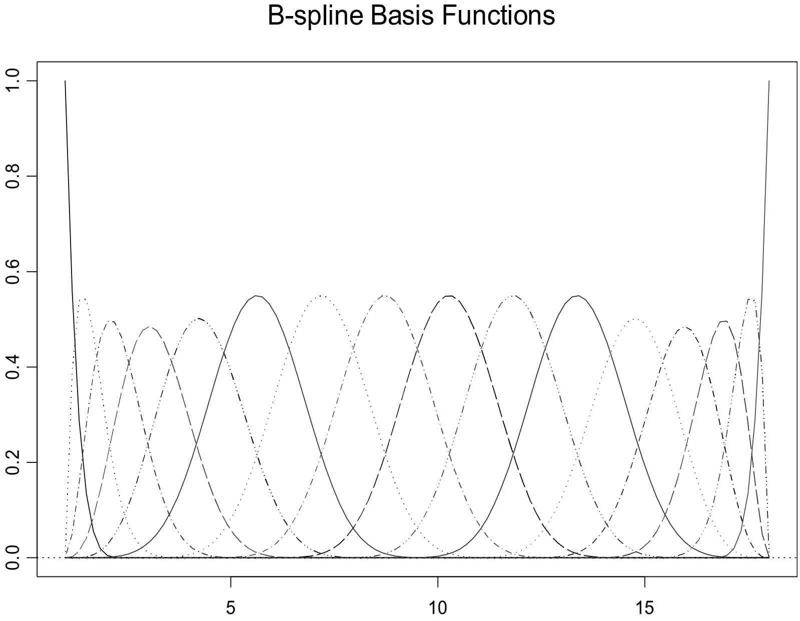

Figure 1.2: *A set of 16 B-spline basis functions.*

Now that we have defined a basis, we need to calculate the coefficients for each height curve. Since there are 93 subjects in this dataset, there should be 93 sets of coefficients (one set for each function). The S+FDA function `fVector` takes the basis, the data matrix, and the independent variable, and returns an object of class "fVector" containing the linear least-squares estimates of the coefficients. An "fVector" object has two additional attributes:

"`basis`", which stores the basis used in the fit, and "`fNames`", which stores labeling information for the data. In the code below, we also specify names for the independent variable (`age`), the subjects (`child`), and the units of the response (`height`). These names are used in the plotting and printing functions.

```
> fHgt <- fVector(object=heightBasis, y=heightData[,2:94],
                fArgs=heightData$age,
                fNames=list(age=heightData$age,
                child=names(heightData)[2:94], height='cm'))
```

Extract the estimated coefficients, basis functions, and function names from `fHgt` using the commands `getCoef(fHgt)`, `getBasis(fHgt)`, and `getNames(fHgt)`, respectively.

Smoothing

Although the basis functions smooth the curves, additional smoothing may be beneficial. The S-PLUS functions for creating functional data objects allow specification of a smoothing penalty in the least-squares objective. The penalty also requires a smoothing parameter, `lambda`. You may estimate an optimal `lambda` by minimizing a generalized cross validation statistic. See section Generalized Cross Validation on page 82 for more details.

Smoothing techniques are largely exploratory in nature, and are discussed in more detail in Chapter 4 of this manual, as well as in Chapter 4 of Ramsay and Silverman (1997). We will have occasion to use smoothing techniques for most of the functional data analysis methods provided in S+FDA.

As an example, penalize the squared second derivative with a penalty parameter `lambda=0.001`:

```
> fHgt2 <- fVector(object=heightBasis,
              y=heightData[, 2:94], fArgs=heightData$age,
              penalty=list(lambda=0.001, linDop=fDop(2)),
              fNames=list(age=heightData$age,
              child=names(heightData)[2:94], height='cm'))
```

Compare with the original data of Figure 1.1 to see how closely the smoothed functions fit the data. The S-PLUS function `fEval` evaluates an object of class "`fVector`" at any point in the domain of the basis. Here, we evaluate the 54 spline curves for the females at the original age values (`heightData$age`), calculate the difference between predicted and observed heights, and then plot the curve differences at the given ages:

```
> hgtFemale<-fEval(fHgt2[1:54], heightData$age)

> plot(heightData$age, hgtFemale[,1], type="n",
       ylim=range(hgtFemale-as.matrix(heightData[,2:55])),
       xlab="Age (years)", ylab="Height Difference (cm)",
       main="Female Height Differences with Splines")
> matpoints(heightData$age,
       hgtFemale-as.matrix(heightData[, 2:55]), pch="o")
```

The resulting plot is given in Figure 1.3.

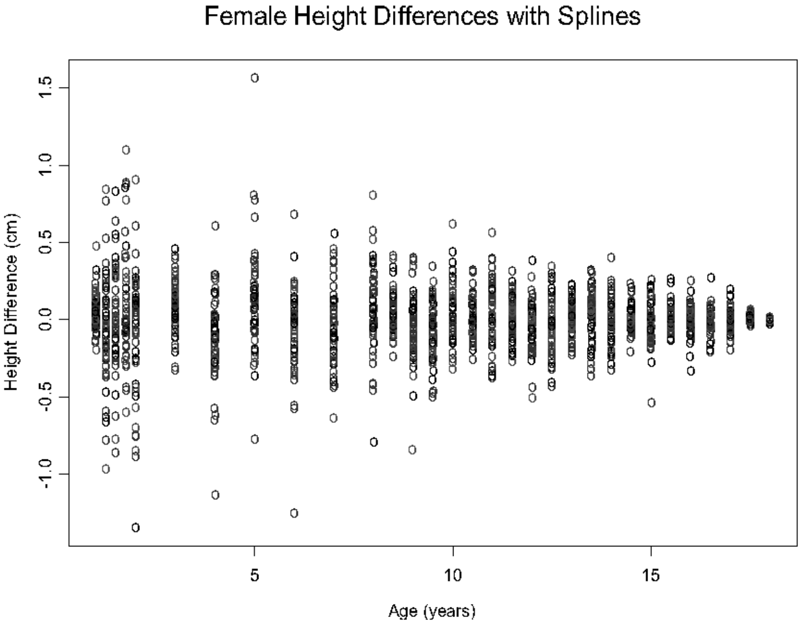

Figure 1.3: *Difference between predicted and actual female height data when using cubic B-splines for function representation.*

The maximum deviation between the spline approximation and the true heights is about 1.5 cm compared with height values of 80 cm or more (see Figure 1.1). These differences are small enough that we consider the smoothed functions to be acceptable for subsequent analysis.

Given a representation of the data as an `fVector` object, it is easy to conduct several kinds of exploratory analyses with S+FDA. Here, we compute the first two derivatives of height with respect to time. We begin with the first derivative:

```
> plot(fVector(fHgt2[1:54], linDop = fDop(1)),
       xlab="age (years)",
       ylab="First Derivative of Height (cm/year)",
       main="Female Height, First Derivative")
```

The result is displayed in Figure 1.4.

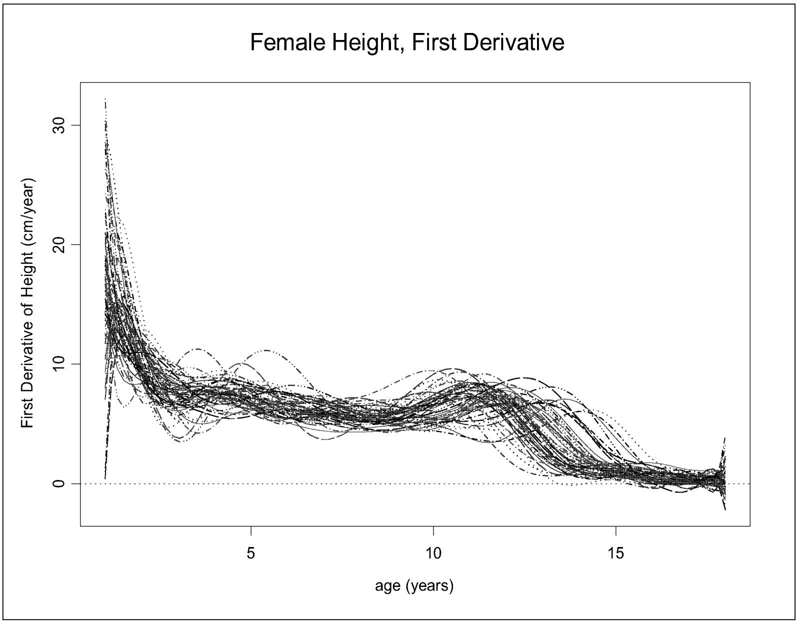

Figure 1.4: *First derivatives of the functional representation of the female height data. The second derivative was penalized for smoothing, with penalty parameter 0.001.*

Despite the large number of curves in Figure 1.4, some general trends are apparent: there appears to be an acceleration in growth around age 4, with a second acceleration after age 10. Further exploratory analysis, such as plotting the mean of the 54 derivative functions, may help reveal more structure.

The plot of the second derivatives is produced in a similar fashion:

```
> plot(fVector(fHgt2[1:54], linDop=fDop(2)),
       xlab="Age (years)",
       ylab="Second Derivative of Height (cm/year^2)",
       main="Female Height, Second Derivative")
```

The result is displayed in Figure 1.5.

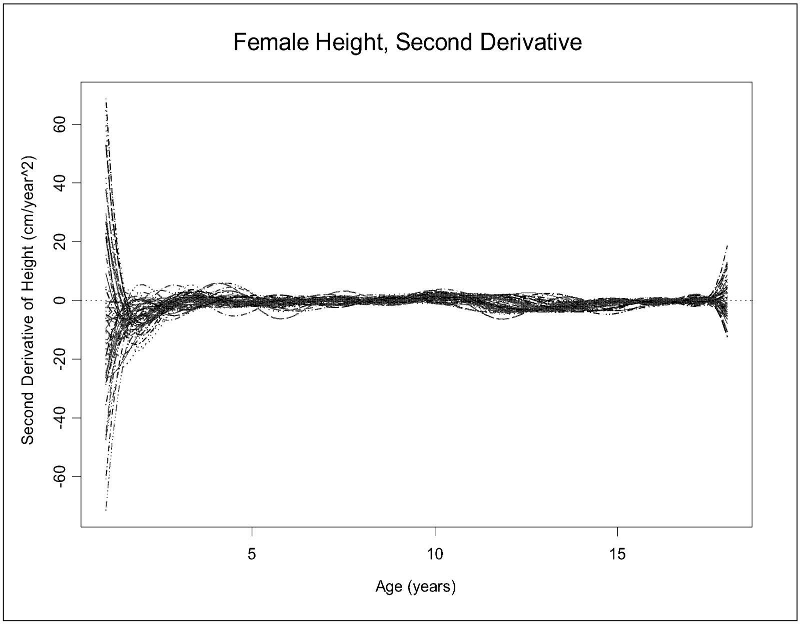

Figure 1.5: *Second derivatives of the functional representation of the female height data. The second derivative was penalized for smoothing, with penalty parameter 0.001.*

The large function values near the endpoints in both derivative plots are due to lack of information concerning values outside the interval. Smoothing by penalizing a higher derivative would reduce the variation at the endpoints, although possibly at the risk of oversmoothing the function. Such considerations are discussed in more detail in the chapter on smoothing.

Because we use splines of degree five (order 6) when fitting the functions, the second derivatives are smooth, cubic splines. Had we fit the raw data with cubic splines (order 4), the second derivative curves would have been piecewise linear. In general, if an analysis requires a smooth `k`th derivative, and smoothness in higher derivatives is unimportant, splines of degree `k+3` (order `k+4`) should be used to fit the functions so that the `k`th derivative will be a cubic spline.

The ease with which you can examine the derivatives is a direct consequence of the functional approach, and one of its main advantages. By regarding the height measurements for each person as a smooth curve, you are no longer constrained by discrete observation times.

A LINEAR MODEL FOR THE HEIGHT DATA

Now consider a functional linear model for predicting sex in terms of the growth rate, the first derivative of the height curve. Since the dependent variable is binary, this model can also be considered a discriminant function for predicting sex in terms of the growth rate.

For the height data, fit a functional linear model as follows:

```
> predLm <- fLM(sex~-1+fVector(fHgt, linDop=fDop(1)),
                data.frame(fHgt=fHgt,
                sex=c(rep(1,54), rep(0,39))))
```

Here the `-1` in the model formula eliminates the intercept, which is already contained in the B-splines. The coefficients in the resulting model are functional. The first coefficient estimate may be plotted as follows:

```
> plot(predLm$coef[[1]], xlab="age", ylab="beta",
       main="Coefficient Function")
```

The resulting plot is shown in Figure 1.6.

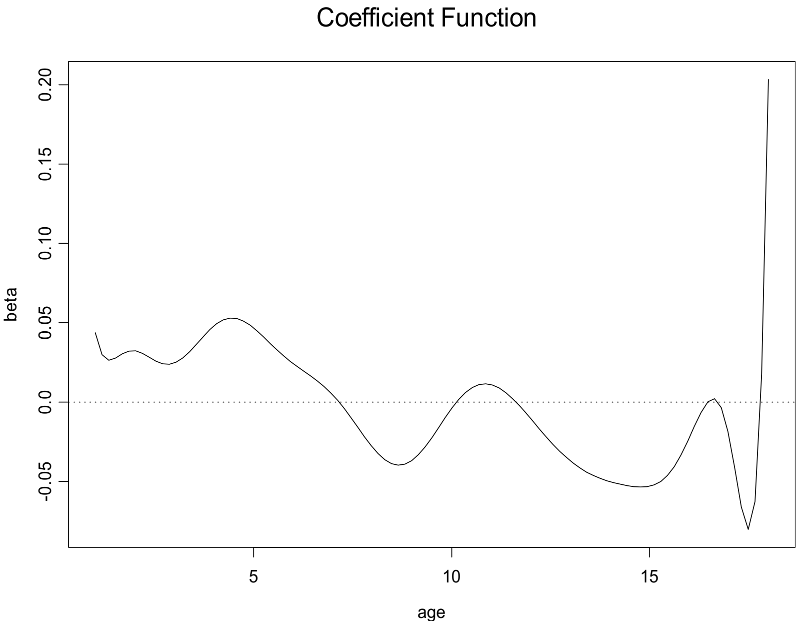

Figure 1.6: *The function of coefficients predicting sex in terms of the height function.*