Dissertation First Draft

190005680

2022-10-10

Contents

1 Introduction

2 What is Functional Data?

To answer this question let's first think about what type of data a statistician is typically tasked with analysing and how this data would be given to the statistician. Let us take an example of a survey where patients are given a drug thought to reduce blood pressure. In this example, the statistician is given blood pressure readings for each patient on the first 4 days after the patient has taken the drug. These readings would typically be stored in a some form of data frame. This data frame could then be converted to a data matrix such that each column of the matrix represented a new patient and each row represented a new day on which readings were taken. An example of such a data matrix (with fabricated data) is given in (1) below.

$$\begin{pmatrix} 142 & 155 & 136 \\ 131 & 134 & 122 \\ 122 & 120 & 110 \\ 95 & 111 & 99 \end{pmatrix} \tag{1}$$

In this data matrix the first entry given in the top left represents patient 1's blood pressure reading (in mmHg) on day 1 after taking the drug, the entry directly below represents patients 1's blood pressure reading on day 2 and so on.

Now consider a scenario where the number of times a patient can have their blood pressure read within the 4 days of the survey was increased to twice a day. If a patients blood pressure was checked twice a day then each row of the data matrix would then represent a reading taken every half day which would make the dimensions of the new data matrix 3×8. A statistician would be in favour of this increase in information as having access to more regular blood pressure readings would allow the statistician to strengthen their inferences about the effects the drug has on a patients blood pressure. The statistician would, however, be quick to point out that this increase would lead to their analysis taking longer as more computational power is required to make inferences from this larger data set. In today's modern era of technology it is becoming increasingly easier for machines to take measurements of various quantities at smaller and smaller increments of time. This leads to the production of massive data sets which can be difficult for even the most complex of computers to analyse. The field of functional data analysis seeks to address this issue.

Returning to the previous example imagine that a persons blood pressure could be taken at any time the statistician requested such that any time within the survey period of 4 days was accessible. This would not be practically achievable as the statistician may ask to have the persons blood pressure read multiple times within the same minute for example. However, if the relationship between time and a patients blood pressure was able to be given by a function with which the statistician could input a point in time within the 4 days and the patients blood pressure would be output then the statistician could do this. Following from this, the statisticians sample would now no longer consist of blood pressure readings at various times for each patient. It would instead consist of 3 functions, one for each patient, which give the relationship each patients blood pressure has with the time since the drug was taken. Assuming that such a function could be found, that would mean that if the observed values given in the data matrix shown in (1) were free from error they would simply be draws from this set of functions for each patient. Assuming these functions could be found we could then consider these functions to be the statisticians new sample of observations which changes the statisticians sample from consisting of data points to instead being a sample of functions. This new sample of functions is called functional data.

This change can also be thought of as expanding the data matrix to now be infinitely dimensional e.g. every point in an infinetly discretised sample of continuous time has an associated row in the data matrix. This expansion makes some theoretical sense as a function is a 1-1 mapping of points in the domain of the function (the input values) to values in the range of the function (the values output by the function). This clearly makes little practical sense as storing infinitely dimensional data matrices is impossible. However, thinking of our samples this way does show some of the advantages that functional data analysis holds over other more conventional forms of data analysis.

3 Advantages of Functional Data

The consideration of the functional data as effectively being an infinitely dimensional data matrix shows a clear advantage of working with functional data over non-functional. This advantage is that it is a potential remedy to the so-called 'Curse of Dimensionality' (Riccio (2022)). This refers to the issue that as the number of dimensions of a dataset increases (be that the number of variables collected for each unit within a study or the number of times that variable is collected within the course of the study) the computational power needed to analyse the dataset grows exponentially. Functional data analysis allows a statistician to not work with large matrices of data but instead with functions for which there are standard mathematical approaches which are more computationally simplistic. High-dimensional data can also lead to over fitting of models to the data which reduces the predictive power of models. Functional data overcomes this issue by simplifying relationships between variables earlier on in the analysis which combats over fitting in the early stages of an analysis.

Working with functions instead of single observations of data also allows a statistician to look at the behaviour of the derivatives of these functions. This can reveal some interesting variation within and between curves that difficult to assess when working with non-functional data. THis advantage will not be discussed at length within this dissertation. The reader can look to (REFERENCE FOR FUNCTIONAL DERIVATIVES)

There is a clear issue with this discussion being that functional data is impossible to have access to in a practical setting. A statistician can only have access to a finite amount of readings of quantities and the relationship between a quantity and time is typically not known otherwise statistical techniques would not have to be used to assess these relationships. In practical applications of functional data analysis the functional data must be estimated from the information that a statistician has access to, being measurements of a certain quantity for several time points. The Creating a Functional Variable section will detail how these functions are estimated from data.

4 Creating a Functional Variable

4.1 How is Observed Data Related To Functional Data?

As discussed previously, if a statistician wants to work with functional data they must estimate these functions from their observed data (which is collections of observations of a quantity over time for several replications). Therefore, there must be a relationship assumed between the functional data and the observed data.

Within this dissertation it will be assumed that the value of the variable of interest that is observed at a certain time t_0 is the value of the function which describes this variables relationship with time evaluated at t_0 with some normally distributed error in the observation. This relationship is better given by Equation (2).

$$Y_i = f(t_i) + \epsilon_i \tag{2}$$

where Y_i is the value of the variable of interest at time t_i and $\epsilon_i \sim N(0, \sigma^2)$ (σ is a constant).

The error in each observation, ϵ_i's, are assumed to be independent and identically distributed normal random variables with constant variance. This assumption is important when we wish to assess the fit of our estimated functional data to our observed data. There are techniques which allow for the relaxation of these assumptions about the error in each observation. ____________ details a method which allows the assumption of independence in errors between observations to relax and _________ details a method which allows a relaxation of normality in the errors.

In order to estimate a functional sample from observed data a process called smoothing must be used. A new example is established in the Basis Functions section which easily displays how a statistician would use this process to estimate a functional sample from a non-functional sample.

4.2 Basis Functions

In order to give the reader an understanding of how to create a functional data object (one function which describes the relationship between a variable and time), both mathematically and in computationally in R, a new example will be used. In this example, it is assumed that the relationship between a variable and time is already known. This variable Y is related to time by the functional relationship given in Equation (3). This relationship is relatively simple and is expressible through typical mathematical notation (non-analytic/non-solvable). The reader should note that this will not typically be the case for relationships between variables and time. In fact, it is generally not possible to attain a simply expressible function for this relationship.

$$Y(t) = f_1(t) = \frac{1}{300}(t^3 + t + e^{-t^2}) \tag{3}$$

Throughout this example, we will conduct our smoothing process as if we do not have access to the analytical form of this function. This function will be the function we are trying to estimate with our smoothing process. It may be clear that we do not have any data to estimate this function from. We will, therefore, simulate some data which mimics the data a statistician may receive in a typical analysis and estimate our function from this data. We will assume that the assumptions established in the How is Observed Data Related To Functional Data? section are true for this data set. We will make the variance of the error terms, (ϵ_i), 30 as this would simulate an observation error similiar to that what would seen in a practical setting. We will simulate our time points to be evenly spaced in the region of $[-40, 40]$. The R Code used to simulate this data is given below.

```r
set.seed(190005680)

function_ex_1 <- function(Time){
  1/300*(Time+Time^3+exp(-Time^2))
}

sigma_2_errors <- 30

Time_Grid <- seq(-40,40,1)

Observations <- function_ex_1(Time_Grid)

Number_Of_Observations <- length(Observations)

Observation_Errors <- rnorm(Number_Of_Observations,0,sigma_2_errors)

ex_1_data <- data.frame(
  Time = Time_Grid,
  Y=Observations+Observation_Errors
)
```

The simulated data is shown graphically in Figure 1 below.

In order to estimate the original function, $f_1(t)$, from this new dataset we need to find some building blocks with which we can constuct this function from the ground up. When working with vectors, basis vectors, sets of vectors that can be linearly combined toconstruct any vectors within a defined vector space, can be thought of as the building blocks for all vectors. Similarly, basis functions can be thought of as the building blocks of all functions.

Basis functions are sets of functions which can be linearly combined to produce any function within the function space they belong to. Basis functions are very powerful tools for function estimation and are often used to estimate functions from sets of points in space. This dissertation will work with two sets of basis

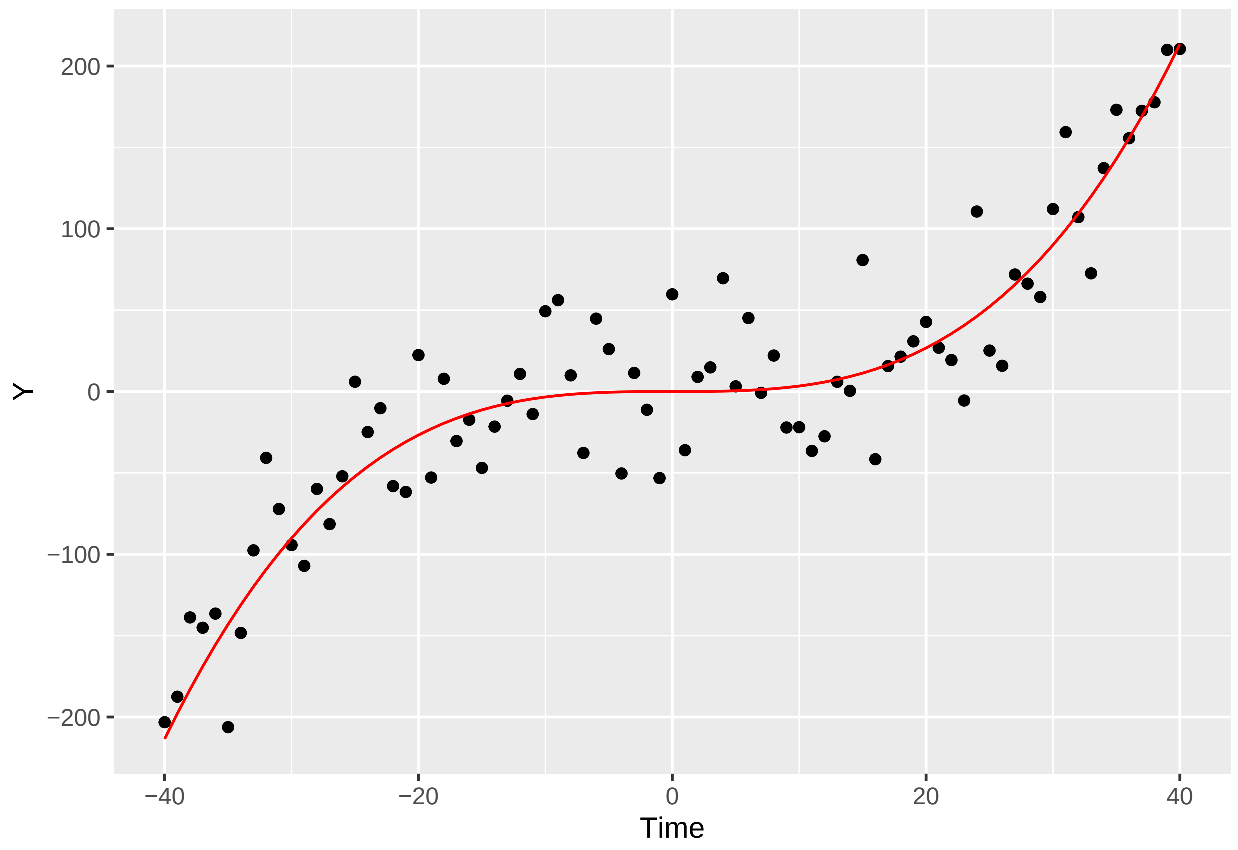


Figure 1: A Plot of Y over Time For Our First Example Dataset. The red line gives the actual relationship Y(t)

functions, the Fourier Basis and the B-Spline Basis which are both sets of functions which can be used to represent functions with a domain in $[t_0, t_1]$ where t_0 and t_1 are arbitary.

We have just established that a linear combination of the basis functions we will be working with can represent functions with domain in $[t_0, t_1]$. Therefore, we can represent our function f_1 as a linear combination of these basis functions. This expansion is shown in Equation (4).

$$Y(t) = f_1(t) \approx \sum_{k=1}^{K} c_k \phi_k(t) = \boldsymbol{c'}\boldsymbol{\phi}(t) \quad t \in [t_0, t_1] \tag{4}$$

where c_k is the coefficient of the k_{th} basis function $\phi_k(t)$ and $\{\phi_k(t) : k \in \mathbb{Z}^+\}$ is a set of basis functions whose span is a set of functions with domain in $[t_0, t_1]$. This is generally true for any function that a person may attempt to estimate.

$\boldsymbol{\phi}(t)$ is a vector of the basis functions used to construct $f_1(t)$ evaluated at time t and $\boldsymbol{c}$ is a vector of coefficients for each of these basis functions.

The reader may notice the approximation sign in Equation (4). For this to be an equals sign K would have to be equal to the number of basis functions which define the function space that $f_1(x)$ belongs to. This is not always achievable as sometimes there is an infinite number of these basis functions in a functional basis. A computer simply could not store all of the coefficients for these basis functions. Therefore, the statistician must select how many basis functions they wish to use based on how well they wish to approximate the function.

It was mentioned that this dissertation will use the Fourier Basis and the B-Spline basis. These will be defined in the The Fourier Basis and The B-Spline Basis sections respectively.

4.2.1 The Fourier Basis

The Fourier basis is an orthonormal set of basis functions which can approximate any periodic function. These basis functions are defined by their period T. The period of a function is defined in Equation (5). The set defined by T can be linearly combined to give any T-periodic function.

$$\phi(t + T) = \phi(t) \tag{5}$$

The form of the functions within the Fourier basis system defined by period T is given below:

$$\begin{aligned}
\phi_1(t) &= 1 \\
\phi_2(t) &= sin(\omega t) \\
\phi_3(t) &= cos(\omega t) \\
&.... \\
\phi_{2n}(t) &= sin(n\omega t) \\
\phi_{2n+1}(t) &= cos(n\omega t)
\end{aligned}$$

where $n \in \mathbb{Z}$ and $t \in \mathbb{R}$.

The ω in these functions is defined in Equation (6).

$$\omega = 2\pi/T \tag{6}$$

To create a Fourier basis system in R the create.fourier.basis command in the fda R package is used. An example is given below:

```
create.fourier.basis(c(45,65),65,5)
```

This command will create the a set which contains the first 65 fourier basis functions with period 5 as defined above. The fda package does not allow for these functions to be defined over all $t \in \mathbb{R}$ as this is not what is typically needed in the context of estimating functions from observed data. Therefore, this function makes the user input a domain that they wish these functions to be defined over. In this case the first argument passed to this function gives this range. For the functions created by this command $t \in [45, 65]$.

The second argument passed to this function gives the desired number of basis functions. Fourier basis functions must always come in sin cos pairs e.g if $\phi_2(t) = sin(\omega t)$ is included in a set then $\phi_3(t) = cos(\omega t)$ must also be included. The function create.fourier.basis accommodates for this, so that if an even number of basis function (K) is requested (there should always be an odd number) the function will also include that basis functions cosine pair (making the number of output basis functions $K + 1$).

The final parameter passed to this function gives the period of the basis system (T). In this case the period of the functions T is 5.

Some examples of periodic data are air temperature over the course of a few days, as the temperature may follow similar trends each day (so period $T = 1\ day$) or the angle a lever is in relation to being upright over time (The fourier basis is often used to estimate functions from data related to angles).

We now move onto defining the B-Spline basis system which is used to estimate functions from data which is not periodic in nature.

4.2.2 The B-Spline Basis

The B-Spline basis system are sets of piece-wise polynomial functions, defined by the range that they are defined over, which will be referred to as $[a, b]$ where $a, b \in \mathbb{R}$, their knots and break points and their order (which will be referred to as n). A set of B-Spline functions which define a basis can be linearly combined to give any spline function of the same degree (which is $n - 1$). A spline function is a piece-wise polynomial functions which is defined over a constrained domain $[a, b]$ that takes values within the real numbers. The most important aspect of spline functions for estimating functions from observed data is that these functions can be combined to give any smooth curve, within $[a, b]$, that is differentiable $n - 1$ times. For example, a spline of order 2 , e.g. $n = 2$, is only differentiable once and, therefore, spline functions of order 2 will be functions that are made up of piecewise straight line segments. Therefore, a set of order 2 B-Spline basis functions, defined on $[a, b]$, can be linearly combined to give any function which is made up of piecewise straight line segments. For a detailed explanation of how B-Splines are defined see the Appendix Section of this dissertation. We can use these sets of B-Spline basis function similarly to the sets of Fourier basis functions to estimate functions from our observed data using the result given in Equation (4).

It is not necessary to know how B-Spline functions are defined to make use of them. The only knowledge necessary are their key features. It was mentioned above that the sets of B-Spline functions which make up a basis system are defined by their knots and break points. The break points of a B-Spline basis are the points where each of the piecewise polynomial sections of the spline (created by linear combination of the B-Splines) meet. The knots are related to the break points such that there is at least one knot at each break point. Each knot placed at a break point signifies how many derivative terms should appear to have a smooth join at a break point. If there is one knot placed at a break point then the first $n - 2$ derivatives should appear smooth across the break point. This is a specific example of the general relation that if there are r knots at a breakpoint then the first $n - r - 1$ derivatives of the spline function must appear smooth across the break point.

It is also important to understand the relation between the number of knots and the number of functions in a B-Spline basis. This relation is given in Equation (7).

$$number\ of\ basis\ functions\ =\ order\ +\ number\ of\ interior\ knots \tag{7}$$

The number of interior knots refers to the number of knots that are defined which are not on the boundaries of the domain in which the basis functions are defined.

This understanding of knots and break points is very useful if a statistician plans on using the derivatives of functions in his sample for analysis as the statistician can define the splines to be smooth curves across the derivatives that he wishes based on this relation. It is also useful to understand how the number of basis functions is related to the number of knots as the number of basis functions inadvertently defines the dimensionality of our new samples.

4.2.3 The Dimensionality of Samples of Functional Data

We have now introduced the concept of both of the basis systems which will be used within this dissertation to estimate functions from observed data. One of the main advantages of functional data analysis is the reduction of dimensionality within the data used for analysis. Returning to the original blood pressure example, the original dimensionality of the data, given in the data matrix in (1), was that there were 4 observations of blood pressure for 3 different replication, those being each patients, overall giving 12 observations. One would assume that when the sample moves from being the observed non-functional data to the sample being 4 functions that the dimensionality of the sample would be 4 the four functions, one function for each patient. This is not the best way of thinking of the dimensionality of the sample. It is better to think of each function within the sample as being defined by the coefficients of the basis functions used to estimate the functions. Typically each function within the sample will be estimated using the same basis system as the data will be similar across all replications. Say we use a set of 2 functions from a B-Spline basis system to estimate the functions which gives the blood pressure of each patient at a given time, then the dimensions of our sample becomes 2 coefficients for each function with a function for each of the 3 patients which gives 3 pairs of coefficients (6 coefficients overall) which define our sample. This is how working with functional data reduces the dimensionality of our samples and this reduction of dimensionality allows computers to conduct analysis faster than when working with non-functional data. This main advantage of working with functional data is important to remember when deciding how many basis functions we wish to use to estimate our functions. Using more basis functions that observations leads to an increase in the dimensionality of our data but using too few could lead to a bad estimation of our function. How do we choose how many basis functions to use?

4.2.4 How Many Basis Functions?

It may seem intuitive to use as many basis functions as is computationally possible to estimate functions from non-functional data. Using more basis functions would allow for our basis function expansion, given in Equation (4), to approximate the shape of the actual function better. While this is generally the case it is not practical or necessary. Often adding more basis functions does not give the statistician any more information about the actual nature of the processes which produce the data. It can also lead to the problem of over fitting as in a practical application the function we do not know the true complexity of the function we are to estimate. The process of smoothing which will be introduced in the Finding Coefficients section will address this issue of over fitting and it is also part of the reason that adding more basis functions often does not produce a greater fit to the actual functions we are trying to estimate. Often it is a case of trial and error to find the number of basis functions which give the desired complexity for the data you are working with. (READ MORE INTO THIS)

4.2.5 Making a Set of B-Spline Basis Functions in R

To render B-spline basis functions in R for use in FDA functions we need to use the create.bspline.basis function from the FDA package. To render the basis system specified in the example given in the appendix we would use the command:

```
create.bspline.basis(c(0,4),norder=2,breaks=c(0,1,3,4))
```

The first argument passed to this function gives the range that the spline basis should be defined over, the second argument defines the order of splines we wish to render and the final argument gives the values that break points should be placed at. Rendering basis functions using break points places one knot at each break point.

The following code gives another example of rendering basis functions. This time the set of basis functions is specified by giving the number of basis functions in the set and the order of the basis functions.

```
create.bspline.basis(c(0,10),16,7)
```

This code will render 16 B-spline functions of order 7 over the range of $t \in [1, 10]$. From the relation given in Equation (7), we can see that there should be $16 - 7 = 9$ interior knots within the basis system and so the functions to be defined by 9 knots that are equally spaced that cover the interval of $t \in [1, 10]$.

Returning to the second example where we are trying to estimate $f_1(t)$ from data we have simulated from the function, the data from this example is non-periodic, e.g. it does not have any repeating patterns, so we would wish to choose B-splines to estimate the function.

A question that arises in relation to our example is how we choose the order of B-spline that we need. The answer to this questions depends on what we are needing to do with our functional data object. If we are in need of it's derivatives then the general rule is to *fix the order of the B-spline basis to be at least two higher than the highest order derivative that is needed*. We are trying to fit a functional data object that we do not wish to take any derivatives of. In this case the general rule is that a linear combination of B-splines of order 4 will provide a smooth enough estimate of the function and so we will fix the order of our set of B-Splines to be 4 for this example.

Returning to our original example, we now know that we want to use a series of order 3 B splines to estimate the function $f_1(t)$ from our observed data as our data is non-functional and we are not wishing to use any of it's derivatives. Our example has 81 data points. Placing a break point at each observation means that we have to have 83 basis functions. This set of basis functions is rendered in R using the code given below.

We now have our set of functions with which we will estimate our function $f_1(t)$ using the approximation given in Equation (4). However, we are yet to find the coefficients of these basis functions in the linear combination. Two methods of finding these coefficients are introduced in the Finding Coefficients section.

4.3 Finding Coefficients

Throughout the previous sections the process of smoothing has been repeatedly mentioned. Smoothing a data set refers to the process of approximating a function from that data set that gives the general trend of the variable of interest over a certain parameter, while excluding errors in measurement. In this case, the parameter a variable is changing over is typically time, however other parameters are sometimes used, e.g. angles. Interpolation has a similar definition however interpolation assumes that there is error in the observations and so the function estimated passes through every point in the data.

We have already stated in Equation (2) that we assume that each observed value within our non-functional data has some error associated with it so we wish to use smoothing techniques to estimate our curves. This section will introduce smoothing two techniques which helps us estimate the coefficients of our basis functions which give the best estimate of our function $f_1(t)$. These techniques will also be used more generally throughout the dissertation to estimate functions when we do not have access to the function which describes a relationship between a variable and time. All methods discussed in the Finding Coefficients section can be applied for any choice of basis function, even choices not mentioned within this dissertation.

4.3.1 Regression Smoothing

The first method of finding the coefficients of our basis functions that we will consider is smoothing using regression analysis.

This is the simplest method of smoothing. This method seeks to make the residuals, being the difference between the estimated function and the observed values, as small as possible. It does this by rearranging Equation (2) as shown below and then minimises these residuals.

$$\begin{aligned} Y_i &= f(t_i) + \epsilon_i \\ \Rightarrow \epsilon_i &= Y_i - f(t_i) \end{aligned}$$

It minimises these residuals by minimising the sum of squared errors as defined in Equation (9).

$$SSE(f) = \sum_{j=1}^{n} [\epsilon_i]^2 \tag{8}$$

$$\Rightarrow SSE(f) = \sum_{j=1}^{n} [Y_j - f(t_j)]^2 \tag{9}$$

where the $Y_j \ for \ j = 1, .., n$ are the n observed observations of the variable of interest and $f(t_j)$ is the function we are estimating, evaluated at time t_j (the time that Y_j is observed at).

This SSE can be rewritten in terms of basis functions and coefficients by substituting in the basis function decomposition of the function $f(t)$ given in Equation (4). This substitution is shown below.

$$\begin{aligned} SSE(f) &= \sum_{j=1}^{n} [Y_j - f(t_j)]^2 \\ \Rightarrow SSE(\boldsymbol{c}) &= \sum_{j=1}^{n} [Y_j - \sum_{k=1}^{K} c_k \phi_k(t_j)]^2 \\ \Rightarrow SSE(\boldsymbol{c}) &= \sum_{j=1}^{n} [Y_j - \boldsymbol{\phi(t_j)'c}]^2 \end{aligned}$$

The vector $\boldsymbol{\phi(t_j)}$ gives the value of each of the basis functions evaluated at the observed time points t_j. All of the values of the basis functions evaluated at the time points that each observation was observed at are stored in a matrix $\boldsymbol{\phi}$ (often called the basis matrix). This matrix is structured such that each column of the matrix contains the basis functions evaluated at 1 observation of t, t_j. For example the first row of this matrix contains the values that the first basis function takes for every observed time point. This matrix is an $n \times K$ matrix.

In the example we have been considering where we are trying to estimate $f_1(t)$ from our simulated data stored in the data frame `ex_1_data`, we can get the $\boldsymbol{\phi}$ from our data by using the eval.basis command. The first argument passed to this function is the time points which the user wishes to have their basis evaluated at and the second argument supplied is the basis functions, created using the create.fourier.basis or create.bspline.basis within this dissertation. The call to this function which gives the $\boldsymbol{\phi}$ matrix for the data in our example is given below.

```
eval.basis(ex_1_data$Time,basis_ex_1)
```

Minimising our SSE function may seem quite difficult. This is a multivariate optimisation problem where there are many covariates to optimise. Taking our example as a case study, the number of coefficients we have to estimate is 81 as we have 81 basis functions and these coefficients can take any number in the real number line.

However this optimisation problem is deceptively easy. In typical non-functional regression analysis the sum of squared errors of the residuals are alos minimised to get regression coefficients. In the case of estimating our coefficients we can treat our $\boldsymbol{\phi}$ matrix similarly to the way one would treat a design matrix, $\boldsymbol{X}$ in a non-functional regression problem.

The SSE of the errors, as a function of the coefficients of the coefficients of the regression parameters, is given in Equation (10).

$$SSE(\boldsymbol{c}) = \sum_{j=1}^{n} [Y_j - \boldsymbol{X'c}]^2 \tag{10}$$

The estimate of $\boldsymbol{c}$ that minimizes this SSE is given by:

$$\hat{\boldsymbol{c}} = (\boldsymbol{X'X})^{-1}\boldsymbol{X'Y} \tag{11}$$

When minimizing the SSE for finding the functional coefficients we can simply exchange our design matrix, $\boldsymbol{X}$, with our basis matrix, $\boldsymbol{\phi}$, to get the estimate of the coefficient vector, $\hat{\boldsymbol{c}}$, given in Equation (12). This vector of coefficients is the vector of coefficients then used to estimate the function.

$$\hat{\boldsymbol{c}} = (\boldsymbol{\phi'\phi})^{-1}\boldsymbol{\phi'Y} \tag{12}$$

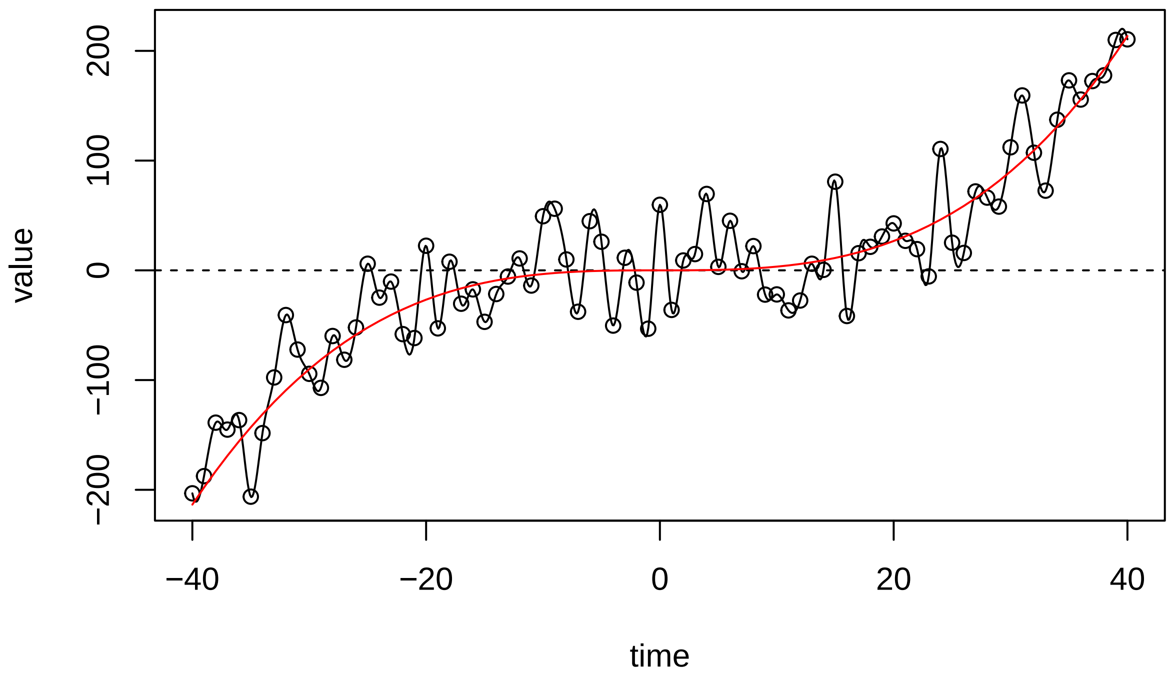
The FDA package has a function that can carry out this calculation for us. We need to give it our basis matrix, $\boldsymbol{\phi}$ and our vector of observed values $\boldsymbol{Y}$ and it will output our coefficient vector.

```
phi_mat_ex_1 <- eval.basis(evalarg=ex_1_data$X,basisobj=basis_ex_1)

lsfit(phi_mat_ex_1,ex_1_data$Y,intercept = FALSE)$coef
```

These functions are not commonly used, however, as there is a function called *smooth.basis* which carries out the process laid out previously using these functions and outputs an already fitted functional data object. To get this function to determine the functional data observations coefficients such that they minimize the SSE we pass the function our vector of observed explanatory variable observations, generally given by $\boldsymbol{t}$ and in our case given by X, our vector of observed values, generally referred to as $\boldsymbol{Y}$, and the basis functions we wish to use to fit the functional data object.

```
## [1] "done"
```



A plot of the fit of this functional data object fit using regression is given below.

This method of finding the coefficients of a functional data object has a few drawbacks. This method does not adjust for how smooth we wish our curve to be and tends to over fit the data if there are significantly more basis functions than data points. This method also tends to give very unstable estimates of the derivatives of the curves near the boundaries of our data which is nit useful as one of the main advantages of functional data analysis is that it allows for analysis of the derivatives of the functional aspects of the data.

4.3.2 Smoothing using a Roughness Penalty

The generally preferred method of finding the coefficients for a functional data object is smoothing using roughness penalties. This method is similar to that used in smoothing using regression analysis in that it uses a the sum of squared errors. However, this method adds a roughness penalty to this sum of squared errors in order to stop the functional data object from over fitting to the data. This sum of squared errors with this new roughness penalty term is shown in Equation .

$$F(\boldsymbol{c}) = \sum_j [y_j - x(t_j)]^2 + \lambda \int_{t_0}^{t_1} [Lx(t)]^2 dt$$

where L is a linear differential operator (which will be discussed in the next section), λ is known as the smoothing parameter and $t \in [t_0, t_1]$.

This can be rewritten using the vector expansion $x(t) = \boldsymbol{c'\phi(t)}$ to get Equation

$$F(\boldsymbol{c}) = \sum_j [y_j - \boldsymbol{c'\phi}(t_j)]^2 + \lambda \boldsymbol{c'c} \int_{t_0}^{t_1} [L\boldsymbol{\phi}(t) L\boldsymbol{\phi}'(t)] dt$$

There are two unknowns in $F(\boldsymbol{c})$ those being our linear differential operator L and our smoothing parameter λ and these are chosen based on what features we wish our functional data object to have.

4.3.2.1 Choosing an Appropriate Linear Diffferential Operator

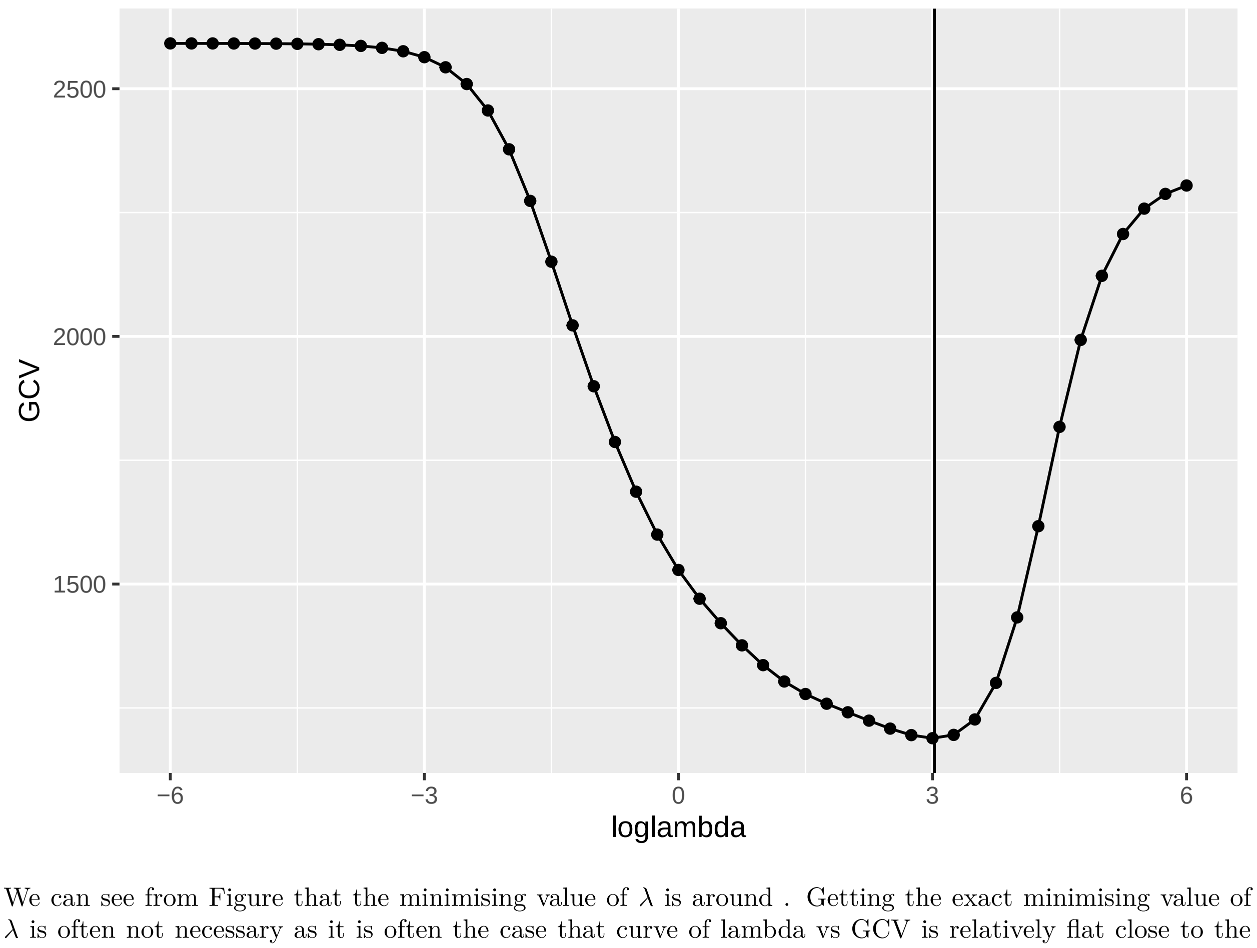
4.3.2.2 Choosing an Appropriate Smoothing Parameter λ The generalized cross-validation (GCV) cruteria was developed by Craven P. and G. Wahba to assist in finding the most appropriate value of λ to use in smoothing using a roughness penalty.

The GCV for a particular functional data object is defined as shown in Equation (13) :

$$GCV(\lambda) = (\frac{n}{n - df(\lambda)})(\frac{SSE}{n - df(\lambda)}) \tag{13}$$

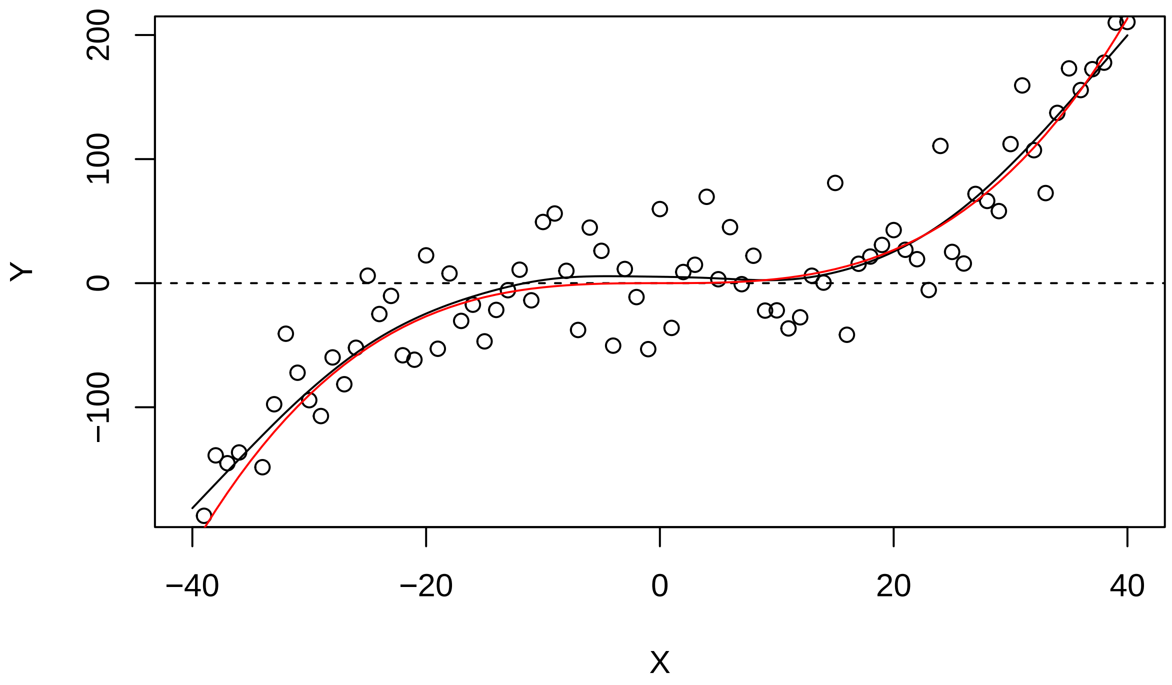
where the SSE is defined as in Equation (9), the degrees of freedom for a particular functional data object are defined as the trace of the $\boldsymbol{H}$ defined in the previous section and n is the number of observations e.g the length of the vector $\boldsymbol{y}$

The value of λ that gives the lowest GCV is the most preferable. The GCV produced by a certain value of λ can be found by accessing the GCV argument of a functional data object in R. An example of how to do this for our example is shown below.



We can see from Figure that the minimising value of λ is around . Getting the exact minimising value of λ is often not necessary as it is often the case that curve of lambda vs GCV is relatively flat close to the the optimal value of λ. Therefore, many values near the optimal value will produce similar results to those given by the optimal value.

```
## [1] "done"
```



```
##
##  Shapiro-Wilk normality test
##
## data:  residuals
## W = 0.98441, p-value = 0.4319
```

5 Appendix

5.1 Derivation of A Second Order B-Spline Basis

To demonstrate the meaning of these terms we will look at the B-spline basis function. B-spline basis functions are defined by a recursion relation. This recursion relation states that for a B-Spline of order K+1, notated by $N_{i,k+1}(x)$ with *knot vector* $U = \{t_0, .., t_k\}$ is defined by :

$$N_{i,k+1}(x) = \omega_{i,k}(x)N_{i,k}(x) + (1 - \omega_{i+1,k}(x))N_{i+1,k}(x)$$

where:

$$N_{i,1}(x) = \begin{cases} 1 & if \;\; x \in [t_i, t_{i+1}) \\ 0 & elsewhere \end{cases}$$

and :

$$\omega_{i,k}(x) = \begin{cases} \frac{x-t_i}{t_{i+k}-t_i} & where\ \ t_{i+k} \neq t_i\ and\ t_{i+k} \in U \\ 0 & elsewhere \end{cases}$$

This recursion relation can be quite difficult to understand for someone unfamiliar with B-spline basis and therefore a worked through example of finding some B-spline functions of order 2 can be found in the appendix of this paper.

There are a few key features of B-splines that are notable. One of these features is that the first and last spline must equal 1 at the boundaries (the first and last knot value). This is because at any points in the domain established by the knots the sum of all the B-splines is equal to 1 such that $\sum_{i=1}^{K} N_{i,k} = 1$. This feature of B-splines means that when a linear combination of these is taken the coefficient of any B-spline function is approximately equal to the value of the B-spline at it's peak. Due to the first and last spline being equal to exactly one at the boundaries the coefficient of these splines is exactly equal to the value of these functions at their peak. Another feature is the relation between the number of knots, the order and number of basis functions. This relation is given by the following:

Say we wish to find a series of B-Spline basis functions of order two and our *knot vector* $U = \{0, 1, 3, 4\}$. Let us first define a B-spline basis of order 1 for the knot vector. This basis would be defined given by the functions:

$$N_{0,1}(x) = \begin{cases} 1 & if\ \ x \in [0,1) \\ 0 & elsewhere \end{cases}$$

$$N_{1,1}(x) = \begin{cases} 1 & if\ \ x \in [1,3) \\ 0 & elsewhere \end{cases}$$

$$N_{2,1}(x) = \begin{cases} 1 & if\ \ x \in [3,4) \\ 0 & elsewhere \end{cases}$$

These functions are shown graphically in Figure 2 below:

These order 1 functions can be used in the recurrence relation to find the splines of any desired order. In this case the derivation for only one of these order two basis functions will be shown but the same method can be followed to find the other two basis functions. We will find the second basis function $N_{1,2}$. First we must start by finding $\omega_{1,1}$. We know $i = 1$ and $k = 1$ (meaning $i + k = 2$) in this case so:

$$\omega_{1,2}(x) = \begin{cases} \frac{x-t_1}{t_2-t_1} & where\ \ t_2 \neq t_1\ and\ t_2 \in U \\ 0 & elsewhere \end{cases}$$

We know $t_2 = 4 \neq 1 = t_1$ so there is only one case meaning:

$$\omega_{1,1}(x) = \frac{x-1}{3-1} = \frac{1}{2}x - \frac{1}{2}$$

Using the same method we find that:

$$\omega_{2,1}(x) = x - 3$$

Therefore, using the recursion relation:

$$\begin{aligned} N_{1,2}(x) &= \omega_{1,1}(x)N_{1,1}(x) + (1 - \omega_{2,1}(x))N_{2,1}(x) \\ &= \left(\frac{1}{2}x - \frac{1}{2}\right) N_{1,1}(x) + (4 - x)N_{2,1}(x) \end{aligned}$$

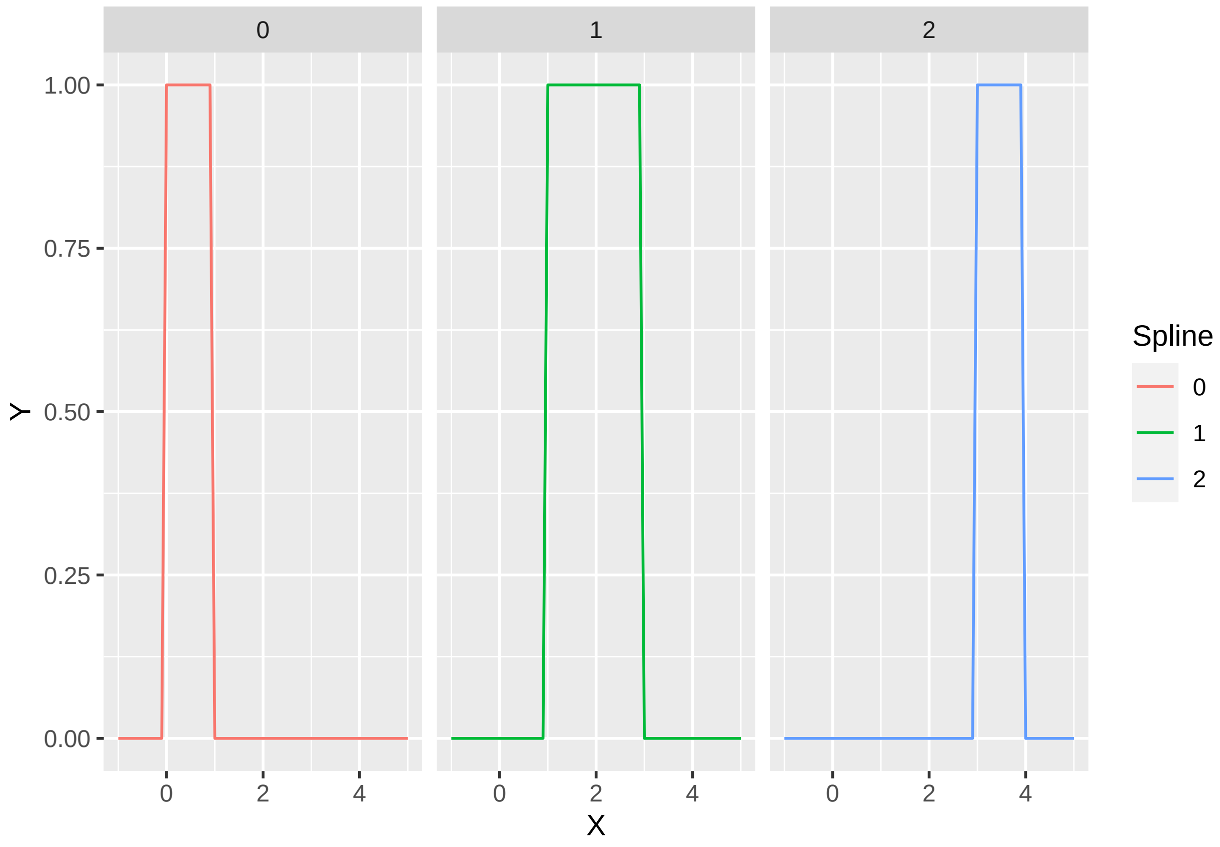


Figure 2: 1st Order B-Splines Represented Graphically

$$\Rightarrow N_{1,2}(x) = \begin{cases} \frac{1}{2}x - \frac{1}{2} & if \;\; x \in [1,3) \\ 4 - x & if \;\; x \in [3,4) \\ 0 & elsewhere \end{cases}$$

In this case we will end up with 3 more basis functions of order 2 those being $N_{-1,2}(x)$, $N_{0,2}(x)$, $N_{1,2}(x)$ and $N_{2,2}(x)$ which are given by:

$$N_{-1,2}(x) = \begin{cases} 1 - x & if \;\; x \in [0,1) \\ 0 & elsewhere \end{cases}$$

$$N_{0,2}(x) = \begin{cases} x & if \;\; x \in [0,1) \\ \frac{3}{2} - \frac{1}{2}x & if \;\; x \in [1,3) \\ 0 & elsewhere \end{cases}$$

$$N_{2,2}(x) = \begin{cases} x - 3 & if \;\; x \in [0,1) \\ 0 & elsewhere \end{cases}$$

Graphical representation of these functions is given below in Figure 3:

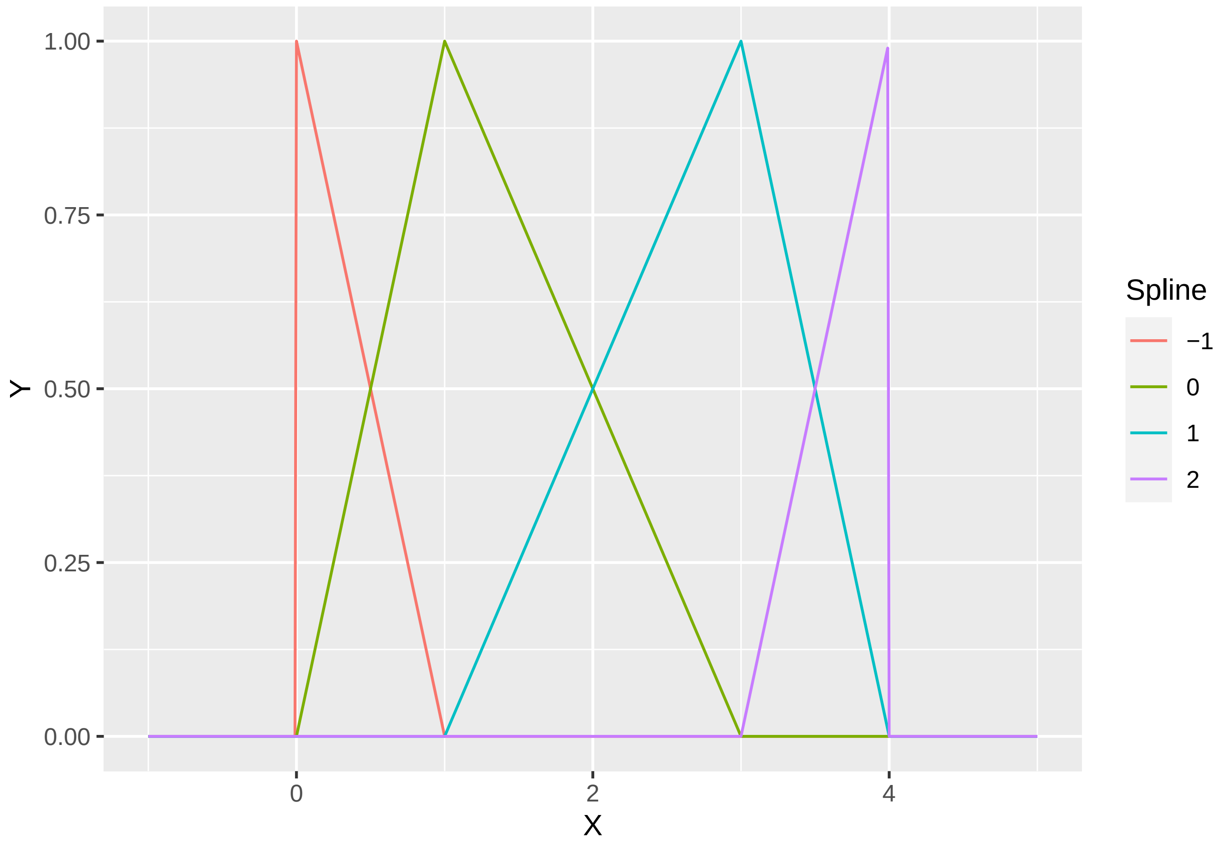


Figure 3: 2nd Order B-Spline Functions Represented Graphically

References

Riccio, Donato. 2022. “Functional Data Analysis: A Solution to the Curse of Dimensionality.” *Toward Data Science*. https://towardsdatascience.com/functional-data-analysis-a-solution-to-the-curse-of-dimensionality-f83dd19fa6e8.