Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Application of Spline and Wavelet Smoothing Techniques For Functional Data

Mohammad Mahdi Eslami

A Dissertation

Presented to the University of Dublin, Trinity College
in partial fulfilment of the requirements for the degree of

Master of Science in Computer Science (Future Networked Systems)

Supervisor: Prof. Mimi Zhang

August 2022

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Mohammad Mahdi Eslami, August 19, 2022

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Mohammad Mahdi Eslami, August 19, 2022

Application of Splines and Wavelets Smoothing Techniques For Functional Data

Mohammad Mahdi Eslami, Master of Science in Computer Science
University of Dublin, Trinity College, 2022

Advancement in statistical technologies has empowered scientists to collect more sophisticated data. This increasing sophistication demands a new conceptualization and consequently, new set of tools and techniques to extract additional information from the underlying patterns of the data. Functional Data Analysis is the study of examining the variability of a dataset when the observations are curves instead of discrete data samples, enabling statistical tools to further study the derivatives of the curve which may contain relavent information to the subject. The first step in Functional Data Analysis is to create smooth approximation functional curves from a set of discrete data points for which some approximation methods are required to describe the behavior of the data. Among the smoothing techniques to create the approximation function in Functional Data Analysis, Splines and Wavelets are of interest in this paper. Splines in nutshell employ a set of knots and fit peicewise polynomial functions between them to provide an estimation function for the data. Wavelets apply smoothing by decomposing the data into set of frequency sub-bands and filter out data samples which do not lie in the specified range. This will remove rapid changes from the data pattern, resulting in a smooth estimation.

The main purpose of this paper is to study the fundamental concepts involved in Splines including different families of splines and different knot placement strategies. Multiple wavelet basis as well as the viablity of each choice is also discussed in this paper. The smoothing factors emplyed by each method to create an estimation function are also studied in this paper by applying the methods different timeseries and spectogramic datasets. This study also provides a practical guide through which these techniques can be applied on functional datasets. An introdcudtion and evaluaiton of various related packages available both in R and Python is included in the practical guide. We implement Splines in R and use PyWavelet library for applying wavelets. Both methods showed competency to be utilized as a smoothing technique for functional data.s

Keywords: Functional Data Analysis, Smoothing, Curve-fitting, Splines, Wavelets, Spline Knot Placement, R, Pywavelet.

Acknowledgement

I would like to extend my gratitude to my supervisor Prof. Mimi Zhang who provided insightful advice throughout the process of writing this dissertaion.

Furthermore, I want to express my appreciation for the teaching staff of the School of Computer Science and Statistics at Trinity College Dublin for their kind support.

I would also thank my family, especially my parents for their unfailing support and love, and to whom I owe a lot.

Table of Contents

TABLE OF FIGURES

TABLE OF TABLES

Chapter 1

Introduction

The rapid advancement in technology has empowered scientists to record more sophisticated data in various contexts including biomedicine, biology, geography, climate, neurosciences and etc. The increasing sophistication of data collection demands a new definition and consequently, new set of tools and techniques to extract additional information from the data. In statistical studies, data is sampled based on a sampling schedule specified by the application requirements e.g. dense, sparse, over a continuum e.g. time, space, wavelength etc. Traditional analysis methods deals with these observations as a set of discrete values whereas modern analysis tools, for instance, Functional Data Analysis (FDA) expresses these discrete observations in form of functions. This enables the analysts to take advantage of the additional information derived from the underlying patterns of the data by evaluating the functions and their derivatives.

The use of differential equations as models and understanding the functional relation between the observations are made possible by the availability of derivatives, adding descriptions of both the dynamical and static characteristics of

underlying patterns. The capacity to quantify these dynamics is a key advantage of Functional Data Analysis over traditional methods. To represent data as functions, the initial step is to creating functional objects from these discrete values. This will be made possible if a curve which approximates the underlying process is fitted to the discrete observations. Next, the functional objects are kept for further analysis and the discrete points are left away. Curve fitting in nutshell is the act of creating the mathematical function or curve that best fits a set of data points, subject to a certain set of conditions. It may either involve smoothing techniques through which a smooth function that roughly matches the data is created or interpolation, which finds a perfect fit to the data points. This makes the smoothing techniques one of the important parts in FDA framework. The smooth function created by the smoothing technique is also referred to as a curve.

Among the different smoothing methods for Functional Data, splines and wavelets are the two techniques discussed in this paper. In essence, splines are a collection of end-to-end polynomials lines joined at points called knots. Each polynomial between knots is adjusted optimally, making splines a powerful tool to capture local behaviors present in the data. As the definition suggests, the number and the location of knots as well the degree of the polynomials are the important factors which control the smoothness of the curve. Knots are frequently set to have equal spacing, However, they may alternatively appear at the determined points of interest. Wavelet Transforms are another strategy which makes use of orthonormal series to address some of the essential limitations of Fourier Transforms. The process of wavelet transformation starts by convolving a basis function, called mother wavelet, with a varying scale and translation factor to the data to extract frequency information. Continuous Wavelet Transform (CWT) and Discrete Wavelet Transform (DWT) are two types of wavelets. In DWT the scale increases discrete values as oppose to CWT which increases continuous values. One of the main applications of DWT is denoising which can result in obtaining a smooth curve.

This paper studies the fundamental concepts involved in spline and wavelet smoothing methods. Different types of each, their required parameters and the implications of different choices are the main topics in this paper. A practical guide on how to apply this techniques on functioanl datasets as well as the evaluation of different programming tools available in this area are also provided. applications.

1.1 Background Research

The phrase *Functional Data Analysis*, was first used in 1991 by Ramsey and Dalzell in there paper in which they discussed the principles of FDA and provided some

tools for applying FDA [1]. Müller has also highlighted the fundamental characteristics of FDA and discusses the core techniques including Functional Principal Component Analysis (FPCA) and Functional Linear Regression in his paper [1]. As the technology advances, more and more sophisticated categories of data are being recorded including high frequency time series, satellite images, neuroimages, biological data and so on. As pointed by Levitin and his fellow researchers, these new categories of data demand new conceptualization and new set of powerful tools to extract information from all the underlying pattern present in the data [2]. To be precise, FDA methods often extract additional information from the data by scrutinizing the function which describes the behavior of the data and its derivatives [2, 3, 4]. This extraction of extra information cannot be normally performed by using traditional statistical methods. Ramsey has explained when data can be treated as a function in his study and the benefits of this approach in more details [5]. FDA effectively considers the entire curve as a single entity, therefore correlation between single sampled observations are unaffected. This indicates a shift in how time series and associated data are handled philosophically [2, 6]. FDA is gaining more attention in a variety of statistical analysis particularly in medical studies. Ullah and Finch have provided a systematic review of FDA applications in various contexts [6]. From a practical standpoint, a measurement error is present in data and every data point is viewed on a discrete grid. Thus, beginning with these discrete observation, the most important step in analysis is to estimate the continuous functional data which describes the behavior of the data. This empower analysts to further examine the derivatives of the functional data. The derivatives of the function usually contain underlying information relevant to the study. As pointed out by many [1, 2, 6, 7], the first step in FDA is to make a smooth approximation for the data. To make functional objects from these data, a smoothing technique should be utilized to provide a smooth estimation which represent the main characteristics of the data pattern. The smooth estimation is also knows as estimation curve [2]. Splines techniques have a long history in creating smooth estimation for functions [8].

Splines in nutshell are a set of polynomial functions joined at mutual points called knots [9, 10]. Splines have been recognized as powerful nonparametric tool in explanatory analysis [11, 12, 13]. This capability of splines has made them a best fit in FDA to generate smooth approximation functions. A comprehensive overview of the smoothing methods used in different FDA applications has been done by [6]. Different choice of polynomial degree, number and the location of knots and some other smoothing factors have created a variety of spline families. The mathematic concepts behind each category of splines has been provided by Carl [8]. Hastie also

in his book gives an intuitive overview of spline families applied on real datasets [14]. Regression Splines and Smoothing Splines comprise the conventional types of splines. Regression Splines control the smoothness of the curve using a component called degrees of freedom which is a function of the knots and the degree of the polynomial fits [8]. The Smoothing Splines however use a high number of equidistant knots and a penalty component to control the smoothness [7, 14]. Consequently, the number and the location of knots in Regression Splines play a key role so as the penalty choice in Smoothing Splines. Many have studied the implication of location of knots in Regression Splines. Paul and Brian in their study compare Truncated Power Series splines with knot placed at Quantiles with Penalized Splines with equidistant knot [15]. William also compared different strategies of knot placement. Some say that the optimal choice of knots is obtained when the knots are chosen based on the characteristic of the input data [16]. A branch of smoothing splines which do not use knot at every point of the data and fir a smooth spline by employing less number of knots are referred to as Panelized Splines (a.k.a P-splines). P-splines use a penalty component to control the level of smoothness. Choosing an optimal value for the penalty is a challenge for Smoothing Splines. The sate-of-the-art is to use generalized cross validation [17, 18, 19]. A good review of packages available in R to apply splines is given by Aris and his fellow coworkers. Spline producers in R are easy to use and provide a variety of options to work with [20].

The choice of smoothing method in FDA should be compatible with the essential characteristics of the data [21]. For periodic data, a useful basis system is the Fourier basis; for non-periodic smooth data with continuous derivatives up to a certain order, a Spline basis are commonly used; and for data with a strong local behavior, an applicable basis system is the wavelet basis [6, 7]. Although these methods are all considered as the smoothing techniques, however, there are fundamentally distinct.

The first literature related to Wavelets dates back to 1901 proposed by mathematician Alfred Haar. The concept was then developed and proposed by Jean Morlet and the term *Wavelet* was invented in 1984 [22]. Wavelet Transforms have been introduced to address some the limitations of Fourier Transforms to provide resolution bot in time- and frequency-domain [23]. Continuous Wavelet Transform and Discrete Wavelet Transform are two categories of Wavelets. Wavelets are mostly used in Signal Processing Context to extract time- and frequency-domain information of the input data [24]. However, Wavelets can also be used as smoothing tool [14, 24]. The term smoothing and denoising are interchangeably used

in Wavelet context. Wavelets Transforms multiply a basis function which is localized in the domain of the data called Mother Wavelets with different scaling factors to the data. Such process is also knows as Convolution [14, 25] . Different choices of Mother Wavelets significantly alter the output of the transform. Many have investigated the effect of choosing different mother wavelets and proposed some criteria to find the best choice for any given data [26, 27, 28].

Thanks to inherent local-adaptivity of Wavelets, they are capable of accepting a broad range of functional forms. Wavelet basis have shown to be particularly beneficial in FDA, particularly when one-dimensional curves are of interest. David and Laura in their work applied Discrete Wavelet Transform on Electrocardiography (ECG) data as the smoothing tool in FDA [29] . Another experiment also made use of Discrete Wavelets on Wind Turbine Data [30] . In essence, when DWT is applied, the data will be decomposed into set of frequency subbands [25] . Then a denoising technique is applied by setting a threshold and removing the frequencies whose their magnitude lies beyond the threshold [25, 29]. That is one of the reasons why Wavelets are a good fit for noisy datasets and data with strong localized behavior. Finding a good compromise for the value of the threshold and an appropriate level of decomposition play key roles in the level of smoothness. There have been studies which investigated the ways through which the best value for threshold and the level of decomposition can be obtained [31, 32].

As things stand, there has been no paper in which a comprehensive comparison between Splines and Wavelets in FDA context is made. In this paper, we review the studies carried out in the two areas and provide practical examples which can contrast Splines and Wavelets in FDA context.

1.2 Motivation

Having reviewed the related studies conducted in this area, it was found out that most of the studies delve into the mathematical aspects behind the methods rather than providing a practical guide on how to utilize them. This could be considered as one of the reasons why smoothing techniques including Splines and Wavelets and in general, Functional Data Analysis are receiving less attention even though they are powerful tool which accommodate most of the requirements of analyzing the modern sophisticated data. There was also no comprehensive analysis found that comprehensively compares Splines and Wavelets in FDA context.

This study is intended to be read both as a review and a manual. That is, we will review the fundamental concepts of Splines and Wavelets and discuss how they can be used in smoothing context in detail. The aim of the study is to provide more

intuitive comparison, rather evaluating the complex mathematical formulas, allowing the readers to focus on the practical side of the experiments. However, the essential mathematical characteristics of the techniques alongside with references to more mathematical treatments for those who are interested are provided.

In addition, a practical guide through which Splines and Wavelet smoothing techniques can be applied on real functional data are also provided. The different options of state-of-the-art programming frameworks which implement these techniques in R and Python are discussed and and evaluated in the practical guide.

1.2 Data Declaration

the set of Functional Datasets that have been used in this are listed in Table (1.1). All the datasets are publicly available on online sources, some of which are built-in datasets available in relevant R packages.

Dataset	*Description*	*Type*	*Source*
Tecator data [33]	*the absorbance values of a piece of finely chopped meet measured at 100 wavelengths.*	*Spectrometric*	*Online*
Queensland Male Mortality Rates [34]	*Age-specific mortality rates for Australia and Australian states*	*Time Series*	*R package*
Sydney Temperature [35]	*Sydney Daily Minimum Temperature from 1959 to 2022*	*Time Series*	*Online*
Triceps Skinfold Thickness Data	*Triceps skinfold thickness of 892 females under 50 years in three Gambian villages in West Africa.*	*Other*	*Online*
Electricity Demand Data [34]	*Half-hourly electricity demands from Sunday to Saturday in Adelaide between 1997 - 2007*	*Time Series*	*R package*
Biscuit Data [34]	*700 point near infrared reflectance (NIR) spectra. The spectral range spans from 1100 to 2498 nm in 2 nm increments for a piece of dough.*	*Spectrometric*	*R package*
El-Nino [36]	*measurements of the water and the atmosphere gathered from a number of buoys scattered over the equatorial Pacific.*	*Time Series*	*Online*
Moisture Data [34]	*Near-infrared reflectance spectra of 100 wheat samples, measured in 2 nm intervals from 1100 to 2500nm*	*Spectrometric*	*R package*
Phenome data [37]	*comprises spectra from*	*Spectrometric*	*Online*
Dublin Bike data [38]	*Bike Availability of stations 2020 - 2022 with 5mintues intervals.*	*Time Series*	*Online*

***Table 1.1 – Functional Datasets**. Functional datasets used in the study with a brief description, category and the source of each.*

1.4 Dissertation Structure

The structure of the rest of the paper is as follows. In chapter 2, we will review the basics of Functional Data Analysis alongside with the examples of such data. Chapter 3 is dedicated to splines and the main features of a spline fit are elaborately discussed. Chapter 4 provides and overview of Wavelet Transforms, different types and how they can be applied in FDA context. In Chapter 5, A practical guide through which the two smoothing techniques can experimented has been provided and the viability of each choice is discussed. In Chapter 6, we discuss the implications, contributions and the future scope of this study. The complete code of each experiment carried out within the sections are available in the relevant appendixes.

Chapter 2
Functional Data Analysis

The advancement in technologies has allowed the statistical studies to collect more sophisticated data. This new sophisticated data demands a new set of tools which are capable of extracting additional information offered by these new types of data. In traditional data analysis approaches, the samples which are collected throughout the experiment are often treated as a set of discrete observations. These observations may also contain natural or sampling noises. Classic statistical techniques can be used on such data, but they are unable to capitalize on the additional information that the underlying functions implies [2] . Functional data analysis (FDA) is a relatively new study that discuses how data which is represented as functions, images or more abstract forms, are analyzed, theorized and categorized. A function is the fundamental unit of functional data, and one or more functions are recorded for each subject in a random sample [1] . The phrase *Functional Data Analysis* was first used in 1991 by Ramsey and Dalzell in there paper in which provide an overview for functional data propose a set of new in FDA [39]. The term function signifies that a functional system is required to apply FDA in applications. This demands a set of tools through which we can obtain smooth estimation functions to represent the main characteristic of the data. This function alongside with its derivates will then be examined to extract additional information from the data which are often not found using traditional data analysis methods.

2.1 Functional Data Object

To define a functional data object, let us review what the basic definition of a function is. In mathematics, a function from a set X to a set Y, assigns to each element of X exactly one element of Y.

The fundamental goal of FDA is to describe discrete observations sampled over a continuum as a function, to create functional data where each function corresponds to a single measurement of the complete measured function [2, 6]. After creating functional data objects, the next step is to bring in modeling and prediction techniques for the next phases of the analysis.

A functional object is not viewed as a single observation. In many cases continuum is time. However, it is conceivable for any continuous domain. A random sample of independent real-valued functions, such as X1(t),..., Xn(t), on a range I = [0, T] form a basic level of a functional data. These data are also known as curve data.

2.2 FDA vs Classical Techniques

In several respects, FDA enhances the capabilities of conventional statistical methods. It is necessary to develop new tools to capitalize on the properties of functional data in order to study the variability of a dataset where the observations are curves rather than points. For instance, the availability of derivatives makes it possible to employ differential equations as models, introducing representations of both the dynamical and static characteristics of mechanisms. The capacity to quantify these dynamics is a key benefit of these approaches over traditional static techniques [3].

In a functional setting, the sampling intervals are not required to be uniformly spaced across all instances and are instead allowed to vary. In other words, the temporal axis may vary in different applications. The data may be sampled densely, sparsely, or neither. Functional data were initially thought of being samples of completely observed trajectories. Later, the idea extended so that functional data objects could be recorded with uniform intervals [1]. That is, if the continuum point for all subjects are:

$$C = c_1, c_2, c_3, \ldots, c_n$$

The temporal grid will be equidistant and for all i and j within the range [1,n] we have:

$$\Delta_c = c_{i+1} - c_i = c_{j+1} - c_j$$

It is the size of this interval which then defines whether the observation is dense or spare. FDA does not assume the values obtained at various points of a single functional object are independent. It also provides tools to deal with data which its samples are correlated and have dependencies to each other [1].

According to Ullah Finch who has provided a comprehensive review of FDA and its applications, FDA has widely been used in diverse categories including weather and climate predictions, satellite images, medical and biomedical studies, analysis of handwriting variations and etc [6] . FDA is still considered a relatively new approach. There are many ongoing studies taking place to make FDA applicable for a wide range of statistical studies.

2.3 FDA Properties

In this section, we will succinctly touch upon FDA properties. The top four features of FDA found in the studies conducted so far are as follows: 1) Smoothing Techniques 2) Dimension Reduction 3) Clustering and 4) Functional Linear Models [1] [6] . The smoothing technique is the topic of discussion in this paper and the latter are not in the scope of this study. However, a brief description of Functional Principal Component Analysis which is used in dimension reduction is provided.

2.3.1 Smoothing Techniques

As mentioned earlier, the sample observations are a set of finite discrete values. Therefore, by directly connecting the samples together without imposing any constraints, the final curve often shows erratic behaviors. This notion necessitates the existence of a smoothing tool to help generate smooth and consistent curve out of the discrete values. This makes the curve become differentiable [1].

In most cases, when we state a curve or a function to be smooth, it is meant that the it should be differentiable to some extent, which signifies that several derivatives can be approximated or calculated from the data [2] . Analyzing these derivatives can be a crucial part of FDA since these derivatives, particularly the first two often have interpretations that are pertinent to the research [40].

When using longitudinal data for hierarchical linear modeling, it is necessary to have a substantial amount of sample points and a high signal-to-noise ratio, characteristics that are not frequently extracted using traditional methods. FDA use derivatives in a variety of ways, and the usage of derivative information is an essential part of FDA [2].

The initial stage in any FDA is smoothing, which contributes in forming the function that describes the data by transforming raw discrete data points into a smooth function. Since it is inventible for a data to appear without inherent or observational noises, it is necessary to reduce measurement or observational errors so that the valid underlying pattern of the data will be emphasized. This also makes the results of the analysis more reliable. In other words, smoothing techniques are also employed to lessen some of the essential randomness involved in the discrete observed data [12, 41].

The findings of Ullah and Finch suggests that, the most utilized smoothing method has been B-spline, mainly due to its flexibility and ease of use [6] . B-splines in essence utilize a set of data points, also known as knots, and piecewise polynomial functions to provide a smooth estimation of the given function. Among other smoothing methods employed in FDA studies, Wavelet basis is also of the interest of this study.

Depending on the underlying behavior of the data being examined, a suitable smoothing technique should be utilized [21] . The smooth estimation must presumably represent the essential characteristics that correspond to those of the original data samples. For instance, Fourier Transforms are typically applied to cyclical or periodic data. Splines, including regression splines, smoothing splines and panelized splines are often applicable on noncyclical and nonperiodic data [6] . Wavelet bases are chosen to represent data displaying discontinuities or rapid changes [7]. We will go through the Spline and Wavelet smoothing techniques in the next chapters.

2.3.2 Functional Principal Component Analysis

After obtaining a smooth curve by applying smoothing techniques, it is time to employ statistical modeling to perform the analysis. The Functional Principal Component Analysis (FPCA) is one of the most popular multivariate analysis techniques for extracting the core features from functional data [6,42]. It is typical to have a high number of uniformly spaced observations for each sample curve when studying a functional data. As a result, in order to describe the key characteristics of a collection of sample curves in terms of a set of uncorrelated variables, a reduction dimension approach is required. This set of uncorrelated features can be later be used in models to perform predictions or extracting other statistical information from the data.

In order to apply FCPA reduction, the data is transformed into a new collection of uncorrelated, ordered variables, where the first few keep the majority of the variance included in all of the original variables. For more detail in mathematics behind dimension reduction see [43].

2.4 Functional Data Examples

Having explained the concepts involved in FDA, it is time to see how real-world functional data are defined. As briefly described earlier, a functional data set can be described as a set of random sample of independent real-valued functions, such as X1(t),..., Xn(t), on a range I = [0, T]. Teactor data set and Mortality rate are two example of real world functional datasets.

2.4.1 Tecator Data Set

Consider the following dataset which is a portion of the original dataset obtained at [33]. The data consists of 100 channel spectrum of absorbances and the amounts of moisture (water), fat, and protein in the meat samples. Spectrometric curve X_i corresponds to the absorbance measured at 100 wavelengths (λ).

$$X_i = (X_i(\lambda_1), ..., X_i(\lambda_j), ..., X_i(\lambda_{100}))$$

Additionally, the fat content of the each meat sample is available ($\boldsymbol{Y_i}$) which was determined using an analytical chemical technique. Table (2.1) shows the structure of the data. As it can be seen in Table (2.1), the entries of the dataset are a set of multivariate functions intaking the wavelength and generating an X function corresponding to that wavelength. The whole Row consisting of 100 observation will then be treated as a single entity in FDA. Figure(2.1) illustrates the Tecator data. Each wave in Fig(2.1) corresponds to a row. X_{30}, X_{12}, X_{200}. More of such data alongside with their description, structure and plots can be found at [44]. Each of the observed curves are then treated as a single entity in FDA.

2.4.2 FDA Package in R

FDS package in R also provides a set of functional data sets including timeseries and spectrogram which can be used in relevant FDA studies [34]. For instance, consider the Queensland male mortality rates (1901-2003). This is a timeseries dataset in which each curve indicates the mortality rate of a specific age from 1901 through 2003. Timeseries are a common class of Functional Data.

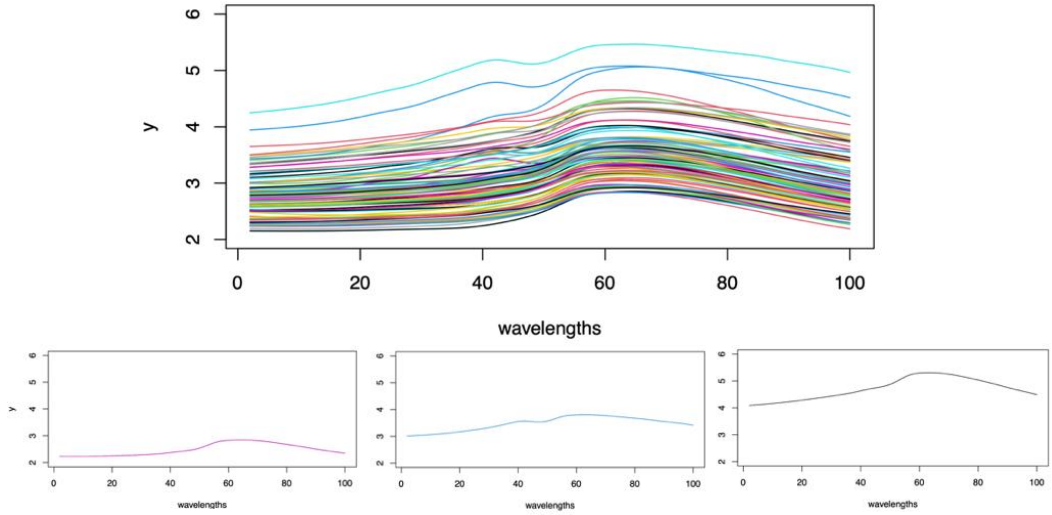


***Figure 2.1 Example of functional data**, Spectrometric dataset. Each wave in (a) represents a functional object, among 215 pieces of finely chopped meat. We observe one spectrometric curve X(i) which corresponds to the absorbance measured at 100 wavelengths. (B), (C) and (D) are samples of functional curves. Original dataset can be found in [37].*

	Col 1	…	Col j	…	Col 100	Col 101
Row 1	$X_1(\lambda_1)$	…	$X_1(\lambda_j)$	…	$X_{215}(\lambda_{100})$	Y_1
⋮	⋮	⋮	⋮	⋮	⋮	⋮
Row *i*	$X_1(\lambda_1)$	…	$X_i(\lambda_j)$	…	$X_{215}(\lambda_{100})$	Y_i
⋮	⋮	⋮	⋮	⋮	⋮	⋮
Row 215	$X_1(\lambda_1)$	…	$X_{215}(\lambda_j)$	…	$X_{215}(\lambda_{100})$	Y_{215}

***Table 1.2 – Structure of Tecator Functional Data**. Each observation of the data corresponds to a wavelength. Each row will create a curve and be treated as a functional object in FDA.*

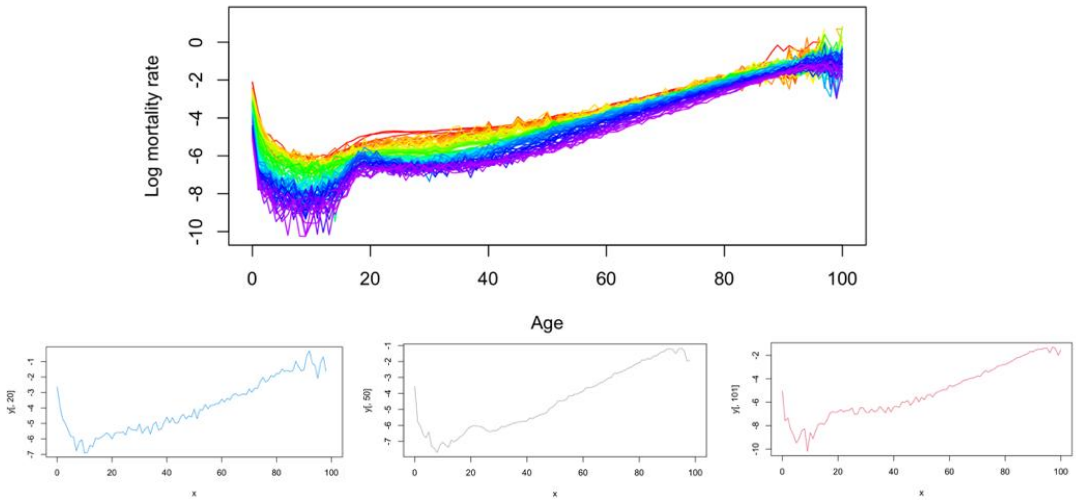


***Figure 2.2 - Example of functional data**. Queensland male mortality rates (1901-2003). FDS package in R contains functional datasets including time series, semi-time series and other types of functional data. Each curve in (A) shows the mortality rate corresponding to a to a particular age. (B), (C) and (D) are the sample functional objects in the data.*

Chapter 3

Splines

It seems appropriate to begin this chpater with the definition of spline and where it comes from. By definition, spline is a a thin wood or metal strip used in building construction or a long thin part that fits into another part of a machine and makes it turn. Historically, craftsmen utilized long thin strips of wood or other material to fair in a smooth curve between prescribed places [10]. It is impossible to use a single piece of wood in order to make curves in the body of a craft e.g. vessels, airplanes. Instead, they would divide the path into small intervals and then use small strips of wood within each to form the curve. To better understand this, suppose that we intend to make a smooth shape as a circle. Apparently, it is not possible to make a circle using a single straight line. Rather, we can divide our circle path into equal sections and draw a straight line to connect the sections with. This will result in an N-side polygon shape while N indicates the number of sections we make. Figure (1) shows how we can arrive at a smooth circle by bridging the gaps between the points using a small lines. The more segments are made on the path, the smoother the final circle becomes. If the number of sides tends to infinity, the polygon turns into a circle without any edges.

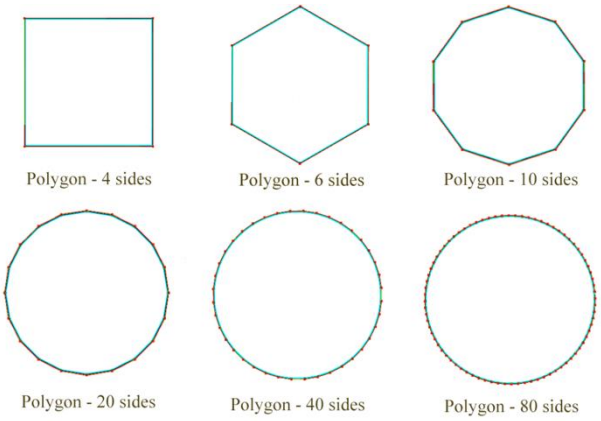


***Figure 3.1 - Transition from a rough shape to smooth shape**. We can arrive at a smooth shape like a circle using high number of straight lines joined on cyclic path. When the number of sides tens to inf, the shape becomes smoother.*

3.1 Interpolation

Splines are also employed in statistical applications to numerically recreate flexible forms. Again, suppose we intend to form a smooth curve to pass through a set of predefined points within the data range. There are multiple ways to draw such curves [45]. Chief among these are Linear Interpolation, Polynomial Interpolation and Spline Interpolation. To give a little context before delving into the splines, we will briefly go through each method mentioned above.

3.1.1 Linear interpolation

Similar to what we did in Fig(3.1), we can use lines to connect the points to each other. However, as expected, since the number of the points are limited, the final curve will be erratic at where the adjacent lines join. This method is called linear interpolation. In practice, linear interpolation takes two subsequent data points, e.g. (x_a,y_a) and (x_b,y_b) while the interpolant at the point (x, y) is defined by the formula (3.1):

$$y = y_a + (y_b - y_a)\frac{x - x_a}{x_b - x_a} \qquad (3.1)$$

Figure(3.2(a)) shows the original data and the linear interpolation. Linear Interpolation takes a few simple steps to compute. However, it is notable that the final curve is not differentiable at the last exterior point due to the

discontinuity. The error in linear interpolation is also proportional to the square of the distance of the points through which we reproduce the curve.

3.1.2 Polynomial Interpolation

Another method in order to interpolate through the points is to replace the linear interpolant with a polynomial of higher degree which fits the data globally. This method is based on the fact that if a data collection comprises n known points, then there exists exactly one polynomial of degree n-1 or less that runs through all the data points. In other words, the target of this method is to find the special polynomial function which defines the relation between the data points. This is the reason why polynomial interpolation works with polynomials of higher degree compared to other methods.

Figure (3.2(c)) shows the polynomial interpolation passing through all the data points. This level of smoothness delivered by the polynomial interpolation comes at the price of finding the coefficients of the polynomial function which fits the data resulting in adding significant computational complexity for polynomials of high degrees. In this case, the polynomial shown on Figure (3.2 (c)) is described by the polynomial line described with Eq (3.2):

$$y = 1.2776 \times 10^{-5}.x^9 - 5.84189 \times 10^{-4}.x^8 + 1.15096 \times 10^{-2}x^7 - 0.120991x^6 + 0.730009x^5$$

$$- 2.54838x^4 + 5.07786x^3 - 6.09975x^2 + 5.35838x - 2.40807 \qquad (3.2)$$

3.1.3 Spline Interpolation

Spline interpolation can be considered as a combination of the above methods and it takes the advantage of both methods to generate a smooth curve that passes through the all data points. However, instead of using a linear function for each interval, spline interpolation uses low-degree piecewise polynomial functions. This adds to the flexibility of the curve as it has more local controls over the intervals between the data points compared to Polynomial Interpolation. This method selects polynomial pieces in a way that they smoothly join together. The resultant curve is then called a Spline.Fig (3.2(d)) illustrates the spline interpolation of the data points. The mathematical spline is formed by substituting the craftsman spline pieces

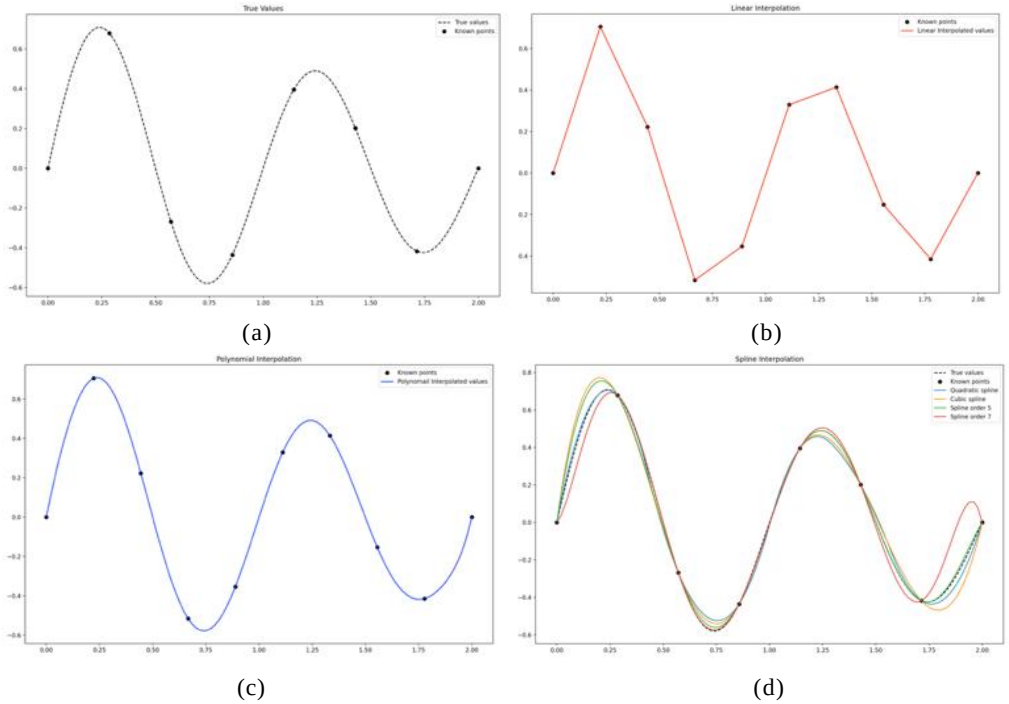


Figure 3.2 – Comparison of Linear, Polynomial and Spline Interpolation. *(a) shows the original data. (b) shows the the linear interpolation. (c) shows the polynomial interpolation. (d) shows the spline interpolation. We can use different degrees for polynomial fits in spline interpolation. It is notable that it may result in lower accuracy especially in the boundary points.*

with smooth functional pieces to mathematically approximate the function with a piecewise low-degree polynomials. The type of polynomial, the number and placement of points (a.k.a knots) is what that defines the type of spline which will discuss it in detail in the future sections.

3.2 Splines for Functional Data

As discussed in Chapter 2, the sample observations of a functional variable are functions that result from the observation of a statistical variable in a continuous argument [7]. Although these observations seem to be continuous, but in essence, they are discrete values that have been sampled in a finite collection of sampling points that may be uniformly sampled or vary between sample units. As a result, the correct functional form of each curve must be reconstructed from a finite number of discrete observations and reconstructing the functional form of the data sample which are discrete observations is the initial step in FDA. In other words, sample observations are not perceived as continuous objects, It is the mathematical efficiency that enable us to perceive these data as samples of curves, surfaces and other

supports fluctuating across a continuum [7]. The primary step in FDA is to turn this discrete recorded data into a fully functional form, allowing each function to be evaluated at any value of its continuous argument [16, 20]. This means that by turning the discrete recorded data samples into a functional curve or surface, we will also be able to estimate the value of the function where the real data sample is not even present.

3.3 Spline, Definition and Properties

To define a spline, we may first need to define its properties.

1. *Basis Functions:* Each spline curve can be represented as a linear combination of basis functions.
2. *Piecewise polynomial:* Each piece of the spine curve is a polynomial function which bridges the gaps between the knots
3. *Knots:* where two polynomial line meet
4. *Continuity and Differentiability:* The level of smoothness of a curve highly depends of the level of continuity and differentiability of the curve.

To statistically analyze the functional data, it is assumed that $f(X)$ represents the curve, meaning that there exists only a single Y value for each X in the domain. The predictor X can be a single variable or multiple variables where $X \in \mathrm{R}$. To produce the spline curve, we first need to define a set of knots in the range on X-space. A spline $f(X)$ will then be a smooth function, passing through the knots and satisfying certain differentiability properties that make $f(X)$ a polynomial line of degree d.

For a spline to be a smooth function, it should meet certain smoothness criterion. Smoothness of a curve is tied with the continuity and the level of differentiability throughout the domain of the funciton. In general, All derivatives of order less than d in a spline curve are continuous. Imposing further constraints to restrict level of differentiability produces different categories of splines. In order to obtain more flexible curves the number of knots or the degree of the polynomial can be increased. There is however a trade-off; increasing the number of knots may overfit the data and increase the variance, whilst decreasing the number of knots may result in a rigid and restrictive function that has more bias [2, 7, 20] .

3.4 Representation of Basis Expansion

To express a function with its basis expansion, it is required to mathematically transform a function into a series or an infinite sum. when it becomes hard to compute a function that cannot be represented with basic operators, utilizing a proper expansion becomes a solution. The expanded representation of the function consists of a finite number of terms, which there sum or production provides the function approximation. The more rough approximation we desire, the fewer terms of the sequence will be employed.

3.4.1 Taylor and Fourier Transform

The most conventional methods among the expansions are Taylor and Fourier transforms. All of the expansion strategies are based on a certain characteristics and information extracted from the original function. For instance Taylor expansion is used to determine the value of the function at every point, provided that the value of the function and all of its derivatives are known at every single point. The Taylor series of function *f(X)* can be obtained using Eq(3.3):

$$\sum_{n=0}^{\infty} \frac{f^n(a)}{n!}(x-a)^n \qquad (3.3)$$

For instance, the expansion of function $sin(x)$ where $x = 0$ will be as below:

$$T(x) = -\frac{x^3}{6} + \frac{x^5}{120} - \frac{x^7}{5040} + \frac{x^9}{362880} - \frac{x^{11}}{39916800} + \frac{x^{13}}{6227020800} + \ldots \quad (3.4)$$

A wide range of Taylor series applications in real world are discussed in [46, 47] . Fourier series are comprised of an infinite sum production of sines and cosines. Due to its periodicity, it is used for analyzing periodic functions [2]. A comprehensive explanation of Fourier series can be found in [48] and a good analysis of its application in MRI data is found in [49].

3.4.2 Basis Expansion

As explained in the previous section, the polynomial interpolation fits a single structure throughout all the data points on *X-space*. It was then discussed that this idea could be extended by partitioning the *X-space* into a series of disjoint intervals. Then, polynomial functions can be adopted within each interval to better capture the local behavior of the function. We explain this notion by giving an example. Figure (3.3) illustrates the minimum temperature of Sydney throughout the year 1859 [35]. The X-space can be portioned into 4 sub-intervals by defining 3 split knots:

$$\xi = (\xi_1, \xi_2) = (100, 200, 300)$$

Given ξ, the simplest model assumes $f(X)$ as a piecewise constant. As a result the complexity of the approximation reduces to estimating the constant value which describes the function. For example, This value could be the sample mean within the intervals. See Figure (3.3 a). This model can be represented with four basis functions where a, b, c and d are the mean value for the range specified in front of each function:

$$H_0^*(x) = 1 \qquad x < 100$$

$$H_1^*(x) = 1 \qquad 100 < x < 200$$

$$H_2^*(x) = 1 \qquad 200 < x < 300$$

$$H_3^*(x) = 1 \qquad 300 < x$$

So that:

$$f(x) = \sum_{i=0}^{3} \beta_i H_i^*(x)$$

The model can be relaxed by utilizing piecewise linear components instead of constant values. Therefore, we require four additional basis functions. Figure (3.3 b) shows how the final curve looks like by applying this model to the data. The method can be extended to arbitrary degrees as shown in Figure (3.3 c). However, by incrementing the degree, new parameters are added which need to be calculated. Having K interior split points, or knots, will create K+1 segment in the *X-space*.

$$H^*_{00}(x) = 1, H^*_{01}(x) = x \qquad x < 100$$

$$H^*_{10}(x) = 1, H^*_{11}(x) = x \qquad 100 < x < 200$$

$$H^*_{20}(x) = 1, H^*_{21}(x) = x \qquad 200 < x < 300$$

$$H^*_{30}(x) = 1, H^*_{31}(x) = x \qquad 300 < x$$

Each polynomial of degree d also has $d+1$ coefficients, resulting in $(K+1)\times(d+1)$ parameters to be calculated. As Figure (3.3) shows, all of the methods resulted in a set of disjoint lines.

Splines are tools to create smooth curves. Having a set of disjoint polynomial lines representing the whole function breaches the smoothness condition. Based on what was discussed in Chapter 2, it is not anticipated that the spline function to be discontinuous at any points. Therefore, we impose some constraints to guarantee the continuity of the spline at where polynomials meet. The first and trivial constraint is that the values of the function at knots should be the same, for example, for a piecewise linear spline, there are three constraints:

$$\xi_1 = 100 : \beta_0 + \xi_1\beta_1 = \beta_2 + \xi_1\beta_3$$

$$\xi_2 = 200 : \beta_2 + \xi_2\beta_3 = \beta_4 + \xi_2\beta_5$$

$$\xi_3 = 300 : \beta_4 + \xi_3\beta_5 = \beta_6 + \xi_3\beta_7$$

It is worth mentioning that it is only required to estimate the free parameters of the systems as the dependent variables can be derived from each other. By adding three constraints, three dependent parameters are removed. Applying the conditions above result the spline curve to appear as Figure (3.3 d). We may also use a basis that directly integrates the limitations rather than having separate basis functions and constraints. For instance:

$$H^*_0(x) = 1 \qquad H^*_1(x) = x$$

$$H^*_2(x) = (x - \xi_1)_+ \qquad H^*_3(x) = (x - \xi_2)_+$$

$$H^*_4(x) = (x - \xi_3)_+$$

Where + denotes the positive part. To add to the smoothness of the curve, apart from using polynomials of higher degrees, we may impose further constraints to ensure the differentiability of the other derivatives. This will be achieved by incorporating additional constraints at the knot locations

- 1^{st}: constraint ensures continuity of f(·)
- 2^{nd} constraint ensures continuity of the first derivative f′(·)
- 3^{rd} constraint ensures continuity of the second derivative f″(·)
- So on.

The first derivative of a function is an equation that provides information about the slope of a tangent line to the curve at any point. If its sign is positive, the curve is rising. If is a negative, it the curve declining. The instantaneous rate of change of the first derivative is measured by the second derivative. The sign of the second derivative indicates whether or not the slope of the tangent line to function is rising and so on for the lower level derivatives. It also provides information about the convexity and concavity of the function at the given entry point. All these information helps us to improve the level of the smoothness of the spline curve. Therefore, for the splines it is necessary for all derivatives of order less than *d* to be continuous. In the Sydney Data, by placing 3 knots and using a polynomial of degree 3 and imposing all the constrains that ensure the smoothness, we expect to arrive at a smooth continuous curve as illustrated in Fig(3.3 (e)).

Having explained the idea of representing a functional curve with a set of basis functions which incorporates all the smoothing constraints, we can say that a general functional spline curve can be represented using the formula below:

$$f(x) = \sum_{K=1}^{K+D+1} \beta_k B_k(x)$$

where the B_k are a set of basis functions described in the previous section and β_k are the associated coefficients. Provided that the degree of the polynomials of the basis function are specified, the only task which needs to be done in order to make an estimation for the function is to estimate the coefficients β_k. Therefore, the complexity significantly reduces compared to

other methods of interpolation e.g. polynomial interpolation. Therefore the estimation of $f(X)$ can be viewed as an linear optimization problem. In other words, spline modelling reduces the estimation of the functions $f(X)$ to the estimation of a small set of real-valued coefficients [2, 20].

3.5 Degrees of freedom

Splines, in essence, are piecewise polynomials linked at knots. The term degree indicates the degree of the polynomials. A polynomial of degree 1 is merely a line and called linear splines. Cubic splines have polynomials of degree 3 and so forth. Degrees of Freedom (DF) specify how many parameters must be estimated to make a smooth fit for the curve data. Degrees of freedom is a function of number of knots and the degree and it varies depending on the type of splines. As the number of knots increases so as the degrees of freedom rise and the spline curve appears wigglier.

The two outermost knots are the called *Boundary Knots*, which are usually positioned at the minimum and maximum of the *X-space*. The other knots are known as *Interior Knots* which are usually the main part of the discussion.

In general, when we partition the *X-space* into some intervals using K knots, there will be $K+1$ intervals created. Suppose we want to connect these knots using a polynomial of degree d. Each polynomial function of degree d, has $d+1$ coefficients. Therefore, for a general spline, the number of parameters that need to be estimated is:

$$DF = (K + 1)(d + 1)$$

However since there are some constrains imposed to ensure the continuity of the derivatives of order less than d, all the variables are not independent. To calculate the independent variables (a.k.a. Degrees of Freedom) we should remove the dependent variables from the total. The derivatives continuity constraints are applied at the knots where two polynomial joins resulting in $K*d$ constrains. Therefore, the degree of freedom for a general spline curve is:

$$DF = (K + 1)(d + 1) - K * d = K + d + 1 \quad (3.5)$$

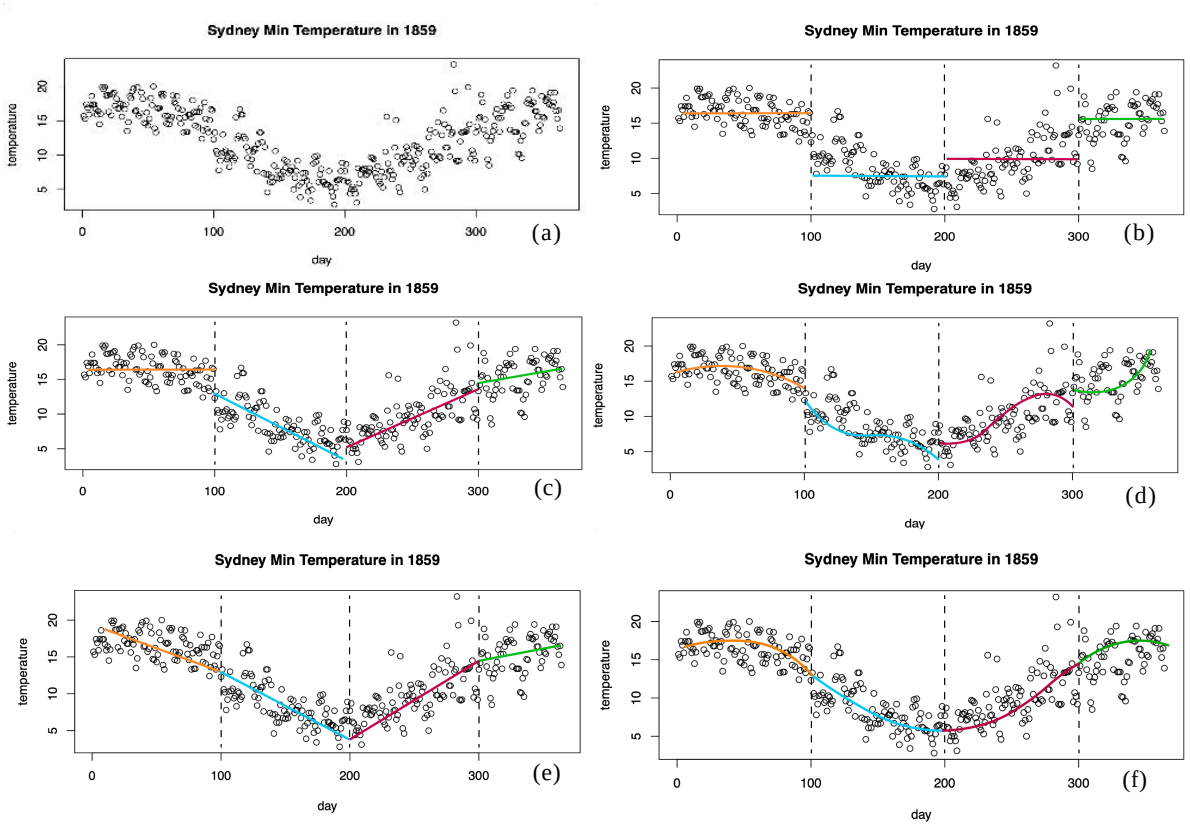


Figure 3.3 – Different Basis Representations *(a) shows the original data which is the time series containing Sydney temperature available at [35] . (b) shows constant basis expansion. (c) shows the disjoint linear basis. (d) disjoint polynomial basis. (e) shows linear basis with continuity, (f) shows polynomial basis with continuity. The main important characteristic of a smooth function is to be continuous and differentiable as shown in (f).*

As shown in in Eq(3.5), the degree of freedom is a function of number of knots and the degree of the polynomial used between the knots.

3.6 Spline Families

Many studies have described the mathematical features of the various spline techniques [8, 19, 50]. There are multiple ways to define the basis functions, and each spline basis has a different set of numerical characteristics [8]. In this Section, we will introduce some of the most popular spline basis, namely the Truncated Power Series, the B-splines, Natural Cubic Splines, Smoothing Splines and P-splines.

3.6.1 Truncated power series

Truncated power series is the most basic form of the smooth splines [8, 20]. As pointed out, continuity means that there are no gaps in the values of a

function $f(x)$. This can be achieved by connecting the polynomial functions together at *knots*. However, there are other continuity constraints which contribute to the level of smoothness and are not necessarily related to the value of the function. These constraints should also be applied to the first derivative $f'(x)$, second derivative $f''(x)$ and so on. Typically, continuity at $f(x)$, $f'(x)$ and $f''(x)$ is enough to make functions look *smooth* to the human eye, but the application may also demand higher orders [20] . The basis function of truncated power series is defined below, built on the representation for continuous piecewise linear model, and increasing the order of the local polynomials:

$$B_1(x) = 1, B_2(x) = x, \ldots., B_{d+1} = x^d,$$

$$B_{d+2}(x) = (x - 1)_+^d, \ldots, B_{K+d+1} = (x - \xi_k)_+^d$$

Where d denotes the degree of the polynomials, K denotes the number of knots and ξ represents the knots vector. + also indicates the positive range of the function. As mentioned in Section 3.5, there are $K+d+1$ basis function used to describe Truncated Power Series. As the formula indicates, it is easy to implement the basis functions of the truncated power series. The first term starts with a basic polynomial of degree d. It will then deviate from the basic polynomial and translating parameters are successively added to the spline function. A truncated power spline is $d - 1$ times differentiable at the knots and has $d+K+1$ degrees of freedom. Figure (3.4) demonstrates the Truncated Power Series basis functions.

Having looked at the basis function, it can be inferred that Truncated Power Series could lead to some numerical instabilities as some of the B_k are defined over the whole range of data resulting in correlations between some basis splines and consequently loosing its support to accurately capture local behavior of the curve [8, 20] . To mitigate this problem, a recursive representation of spline basis functions has been proposed which is called B-spline. The recursive definition of B-spline representation causes the subsequent polynomials to inherit from each other, making the final curve as consistent as possible. The code for generating Truncated Power Series Basis Function can be found in appendix (1).

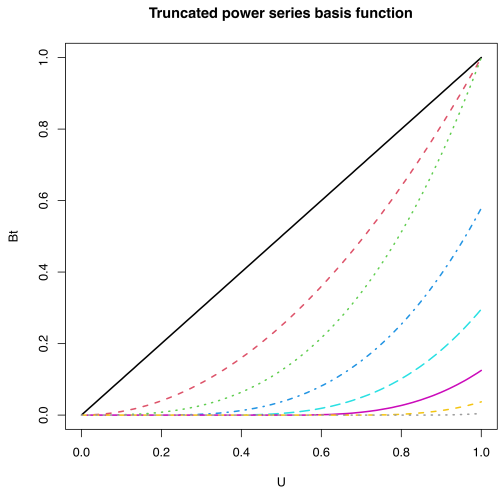


Figure 3.4 – Truncated Power Series Basis Functions.

3.6.2 B-splines

B-splines basis function offer an alternative, computationally convenient representation of basis functions and it is recognized as a powerful smoothing tool which are capable of capturing the local behavior of the data. the recursive formula of B-splines, the knot vector has to be augmented in the boundaries. The general polynomial basis functions are obtained using a recursive relation described below.

$$B_{i,j}(x) = \frac{x - \xi_i}{\xi_{i+j} - \xi_i} B_{i,j-1}(x) + \frac{\xi_{i+j+1} - x}{\xi_{i+j+1} - \xi_{i+1}} B_{i+1,j-1}(x)$$

Where j is the degree and i walks through the knots vector. To help better understand B-splines, let's assume a cubic spline (D=3) on the range x ∈ [0, 1] with three interior knots as:

$$\xi^* = (0.25, 0.5, 0.75)$$

As the first step, consider B-spline with degree 0 which is a series of locally constant functions over the range of *X-space*. Based on the recursive definition of formula, to obtain basis functions with higher indexes, we require to obtain the value of ξ_0 and $f(\xi_0)$. Otherwise, the value of the first basis function cannot be obtained. On the other hand, to obtain the penultimate basis function, we should know the value of the last ξ_{K+1} and $f(\xi_{K+1})$ In advance. Boundary knots define the range in which the spline is

evaluated. That is meant by the augmentation of the knots on boundaries in B-splines. Suppose the values of the augmented boundary knots are as follows:

$$\xi_0 = 0,\ \xi_{K+1} = \xi_4 = 1$$

resulting the knot vector to be:

$$\xi^* = (0, 0.25, 0.50, 0.75, 1)$$

We can mathematically define the B-spline basis functions of degree 0 as:

$$B_{i,0}(x) = \begin{cases} 1 & \xi_i < x < \xi_{i+1} \\ 0 & otherwise \end{cases}$$

boundary knots are meant to ensure that all of the locally-defined $B_{i,0}(x)$ are not ill-conditioned and instead are all well-defined. Figure (3.4 a) shows the basis function of B-spline when $j = 0$. It is notable that each basis function spans over two knots. The construction of the B-spline representation of a linear spline ($j=1$) is obtained by taking a weighted average of the $B_{i,0}(x)$ functions:

$$B_{i,1}(x) = \frac{x - \xi_i}{\xi_{i+1} - \xi_i} B_{i,0}(x) + \frac{\xi_{i+2} - x}{\xi_{i+2} - \xi_{i+1}} B_{i+1,0}(x)$$

The two intervals which include three knots for the linear B-spline are $[\xi_i, \xi_{i+1})$ and $[\xi_{i+1}, \xi_{i+2})$ respectively. As the formula indicates, there are three knots involved in each basis function. Therefore, we need to further augment the knots and also ξ_{-1} and ξ_{K+1} to the knots vector. However, since the value of $f(\xi_{-1})$ and $f(\xi_{K+1})$ are not specified, there are some strategies to manage this problem. The most commonly used strategy is to set the augmented boundary knots equal to the original boundary knots [14, 20] , meaning that the augmented boundary knots prior to ξ_0 will have the same value as ξ_0 and augmented boundary knots beyond the ξ_K will have the same value as ξ_k. Resulting the knots vector of our example to be:

$$\xi_{-1} = \xi_0 = 0,\ \xi_K = \xi_{K+2} = 1$$

$$\xi^* = (0, 0, 0.25, 0.50, 0.75, 1,1)$$

In this strategy the basis function for the augmented knots will be:

$$B_{i,0}(x) = 0 \ \ if\ \xi_i = \xi_{i+1}$$

Since the degree in linear B-spline is 1, each term of $B_{i,1}(x)$ is the product of two terms, a locally constant function and a locally linear function to form a linear equation, resulting the overall basis function to $B_{i,1}(x)$ linear as well. Fig (). The development of the B-spline basis functions of a quadratic spline is likewise based on the recursive relation. As the formula () shows, each basis function spans over four subsequent knots which necessitates further knot sequence augmentation. i.e. ξ_{-2} and ξ_{K+3}. The form of each quadratic spline basis function is obtained by taking a weighted average of the subsequent $B_{i,1}(x)$ functions:

$$B_{i,2}(x) = \frac{x - \xi_i}{\xi_{i+2} - \xi_i} B_{i,1}(x) + \frac{\xi_{i+3} - x}{\xi_{i+3} - \xi_{i+1}} B_{i+1,1}(x)$$

Similar to what we saw in linear B-spline basis functions, each component of Bi,2(·) is the product of two functions. However, since we are constructing a quadratic basis functions, the two components are two linear functions defined on the intervals $[\xi_i, \xi_{i+2})$ and $[\xi_{i+1}, \xi_{i+3})$ respectively. As a result, each product is locally quadratic so as the final basis functions. Fig(). To ensure that each $B_{i,1}(\cdot)$ is well-defined, we are again required to further augment the knot sequence with two additional boundary knots; ξ_{-1} and ξ_{K+2}. As such, $K+d+1=5$ basis functions are defined. By convention, knots beyond the *Boundary Knots* are set to equal ξ_0 or ξ_{K+1}. The same process is applied when we extend the basis function to be a cubic spline. The construction of the B-spline representation of a cubic spline is based on a relation below, and additional knots are added to the knots sequence, namely: ξ_{-3} and ξ_{K+4}. Each basis function of degree *3* is obtained by taking a weighted average of the $B_{i,2}(x)$ functions:

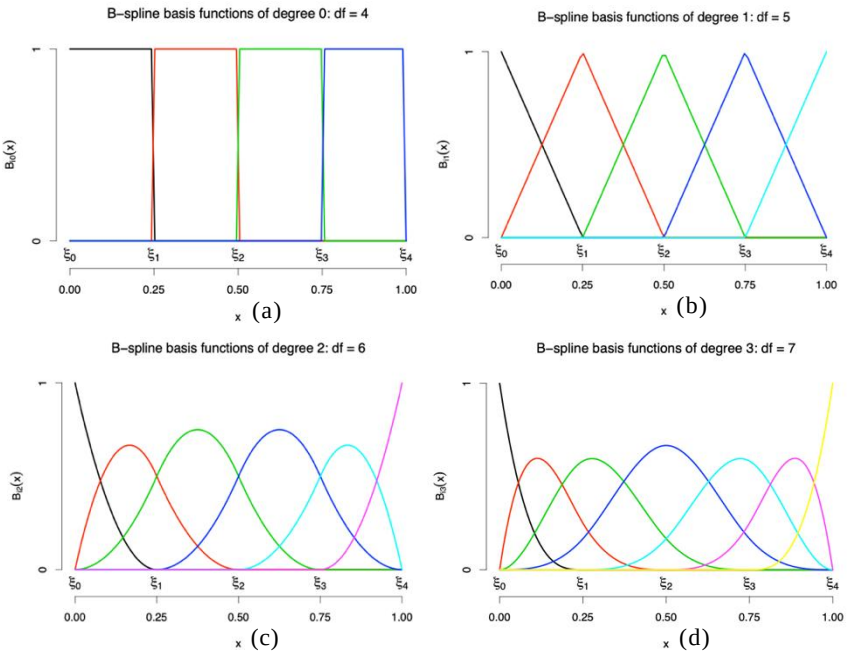


Figure 3.5 – Different B-spline Basis Functions *(a) shows the B-spline with degree 0. (b) shows linear B-spline with 5 degrees of freedom. (c) shows the quadratic B-spline basis with 6 degrees of freedom. (d) shows cubic B-spline basis with 7 degrees of freedom.*

$$B_{i,3}(x) = \frac{x - \xi_i}{\xi_{i+3} - \xi_i} B_{i,2}(x) + \frac{\xi_{i+4} - x}{\xi_{i+4} - \xi_{i+1}} B_{i+1,2}(x)$$

Each component of $B_{i,3}(x)$ is the product of a locally linear and locally quadratic function defined on the intervals $[\xi_i, \xi_{i+3})$ and $[\xi_{i+1}, \xi_{i+4})$ respectively. Consequently, each product is locally cubic, as so the overall basis function. The same process will be applied if B-splines with higher degrees are required.

3.6.2.1 B-splines on Functional Data

Having understood the basic concepts of B-splines, it is time to conduct experiments on some functional dataset and see how one can employ B-splines to fit a smooth curve on the input data and what the implication of different choices of smoothing parameters are. Figure (3.6) demonstrates the effect of increasing number of knots. As discussed in the previous section, by increasing the degree and the number of knots, the B-spline model starts to capture the local behavior of data. From a basic cubic polynomial spline

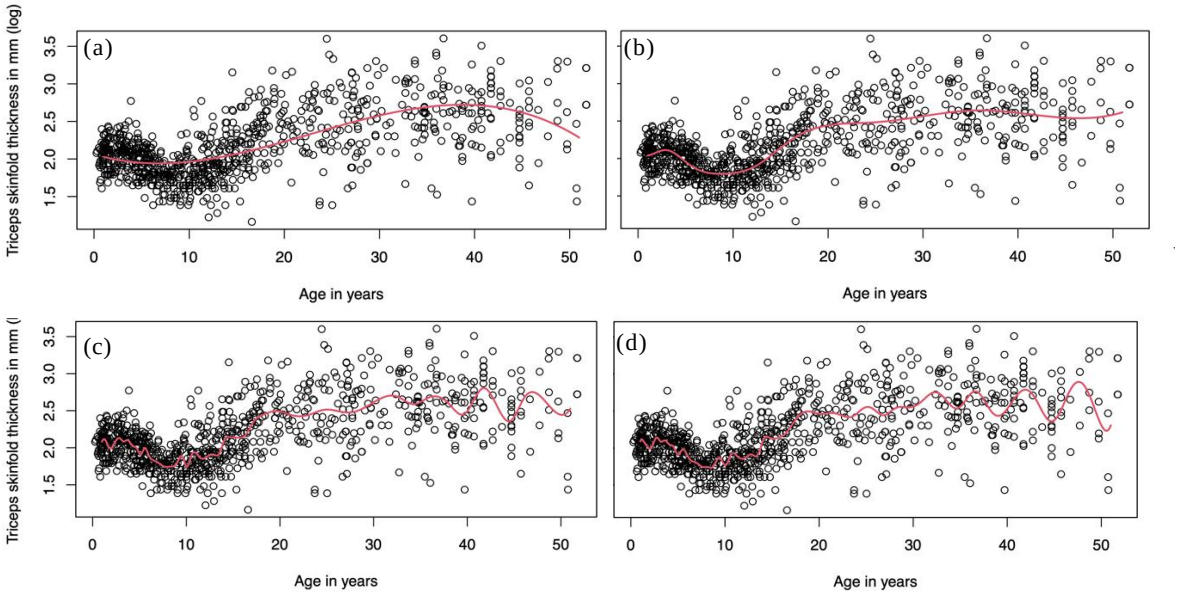


Figure 3.6 – B-spline on Triceps Skinfold Thickness Data *(a) shows the cubic B-spline with df=3. (b) shows cubic B-spline with df=5. (c) shows cubic B-spline basis df=10. (d) shows cubic B-spline with df=20. As the degrees of freedom increases, the spline fit starts to capture the local fluctionas in the data. It imposes a trade-off between accuracy, smoothness and computational complexity Complete code is in appendix (2).*

(top-left) we arrive at much more complex curve which appears to be more promising and locally accurate. Using the same algorithm mentioned above, similar spline fir can be generated for other datasets. Figure (3.6) illustrates a time series data related half-hourly electricity demands in Adelaide between July 1997 and March 2007 on Mondays. The sub charts are splines with 3, 5, 10, 20 number of knots respectively. We can see that the accuracy of the spline model increases as the number of knot increases.

By taking a look at Figure(3.6(g)) and (3.6(h)), we can see notice a spurt in the very ends of the curve in the latter. This is because we have increased the degree of the polynomials splines between each knot from 3 to 7. This causes the curve to act dramatically, particularly on the boundary areas as no prior or succussive knot is present to curb the polynomial of higher degrees. Apart from the boundary areas, there is no major difference observed between (3.6(f)) and (3.6(g)) in terms of smoothness. That is why cubic splines are to be preferred as they are the best compromise between the smoothness and computational complexity. (Code in appendix (3)).

3.6.3 Cubic Splines

Cubic splines are a special case and the most commonly used version of B-splines. As described in the previous section, In cubic splines, cubic polynomial are created in an interval between two successive knots. Since

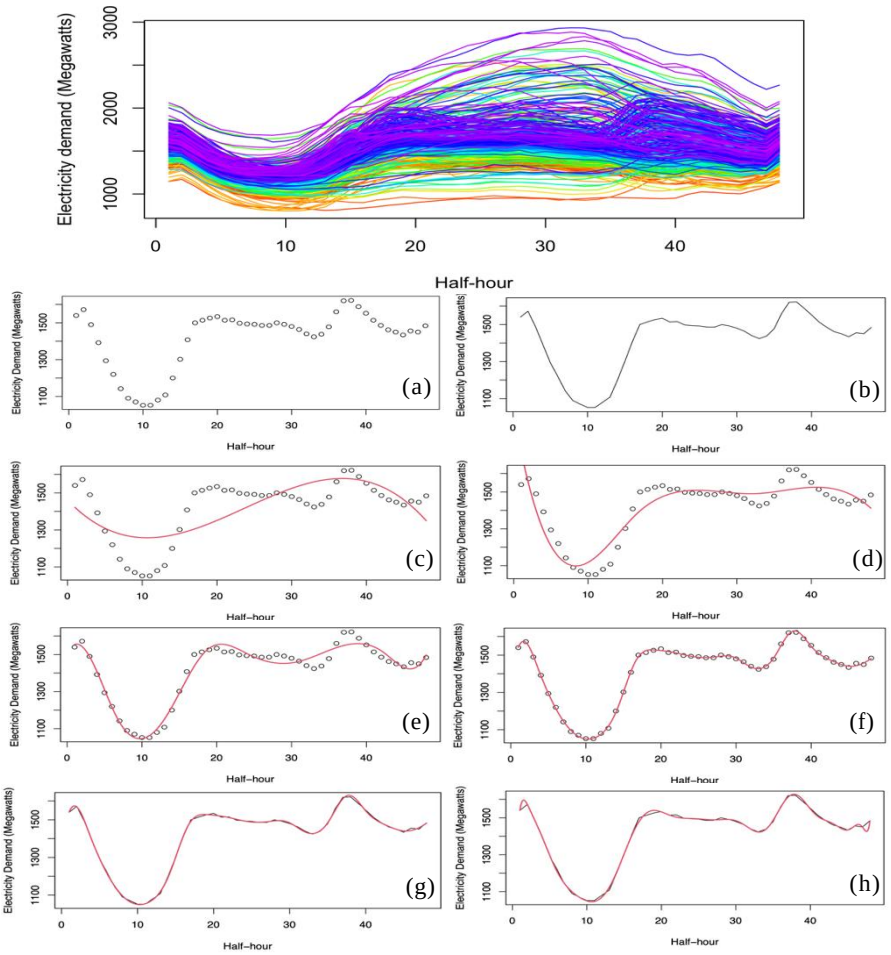


Figure 3.6 – B-spline on Electricity Demand Data. *The data is available in in fds package in R. (a) shows the discrete observations of the data. (b) shows the linear interpolation. (c) shows a cubic B-spline fit with df=3. (d) shows cubic B-spline fit with df=5. (e) shows cubic B-spline basis df=10. (f) shows cubic B-spline with df=20. (g) and (h) compare the result when degree of the polynomial basis rises to 7 instead 3. Note the spurt in the boundry areas in (h) when degree = 7.*

the cubic polynomials are used in the intervals, there will be four parameters on each of the $K + 1$ regions if there are K knots. As discussed earlier, to ensure directionality, each polynomial is d-1 times differentiable minus three constraints for each knot, resulting in a $K + 4$ degrees of freedom.

$$DF = 4(K + 1) - 3 * K = K + 4$$

A cubic spline function, with three knots (ξ_0, ξ_1, ξ_2) will have 7 degrees of freedom and can be written as:

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 (x)^3 + \beta_4 (x - \xi_0)^3 + \beta_5 (x - \xi_1)^3 + \beta_6 (x - \xi_2)^3$$

One can simply produce cubic B-splines in R by setting the degree of the B-spline to 3, which we touched upon that in the previous section.

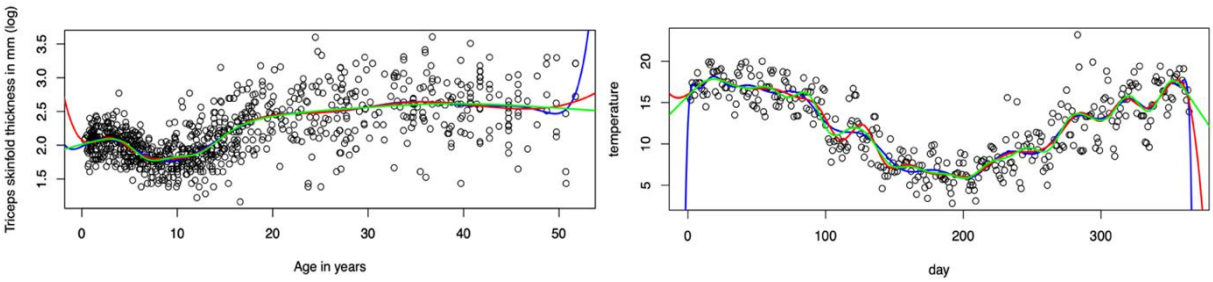
3.6.4 Natural Cubic Splines

Natural Cubic Splines are another class of Splines. The motivation behind creating this spline is that the behavior of high degree polynomials fit to data can be erratic. This problem intensifies as we approach around the boundaries. The boundaries are where we have no further data to fit a polynomial curve. Therefore, continuing the trend of the polynomial curve outside the boundaries could lead to significant roughness. In splines, we apply a polynomial curve between each pair of the subsequent knots. As a result, Beyond the boundary knots, where there is no more knot to connect the boundary knots to, polynomials act much more erratically than the corresponding global polynomials in that particular range.

To mitigate this roughness caused by polynomial curves of high degrees, additional constraints are imposed at the boundaries to control the level of roughness outside the range. Natural cubic splines assume that the function is linear beyond the boundary knots.

By applying this constraint, there are 2 parameters on both boundaries that are removed by turning cubic splines to linear spline, making degrees of freedom decrease by four in total [20, 51] . This can be more effectively invested by scattering additional knots in the inner area [51].To demonstrate this, we augment the range of test data to go a little beyond the boundaries to compare which model can predict more naturally. Fig (3.7) provides a visual comparison of different level of roughness in boundaries in global cubic polynomial fit, cubic B-spline and Natural Cubic Spline.

As it can be observed in Figure (3.7), the most erratic behavior in the boundaries belong to the global polynomial fit. By comparing the behavior of the cubic B-spline fit and the Natural Cubic Splines fit, it is seen that the natural cubic splines has behaved more naturally (linearly), making Natural Cubic Spline a good candidate for extrapolation.



(a) (b)

Figure 3.7 – Comparison of Behavior of Splines on Boundaries. *(a) is Triceps Skinfold data. (b) shows Sydney Temperature data. The blue line is the Global Polynomial Fit of the data. The red line indicates a Cubic B-spline and the green line shows the Natural Cubic Spline fit for the data. As it can be seen, the behavior of the Natural Cubic Spline fit is linear in the boundaries and behaves moderately compare to other methods. B-spline performs better than the Polynomial fit.*

Figure (3.8) shows the effect of different degrees of freedom on Biscuit data [34] . The complete code including how to load the data can be found in appendix (3).

3.6.5 Smoothing and Panelized Spines

So far, we have reviewed some of the Spline strategies to provide a smooth approximation for the function which describe the behavior of the data. These methods are conventionally referred to as *Regression Splines.* Regression Splines utilize a fixed-knot points. Therefore, the computational complexity in regression splines are controlled by the number of knots and the degree of the polynomials used in the intervals. Apart from the basis function, the number and the location of knots significantly impact the behavior and the shape of the final curve. In other words, the choice of basis function turns out to have a little impact on the smoothness of the final curve as they all create curves which appear smooth-enough to human-eye. Thus, there are knots which provide control over the level of smoothness of the estimated curve. A large number of knots indicates considerable flexibility, but it may also result in data overfitting. On the other hand, A limited number of knots may yield a preliminary approximation prone to under-fit bias [7, 20, 52].

One way to avoid dealing with number and the location of knots is to use Smoothing splines. Smoothing Splines were first introduced by O'Sullivan by inserting a penalty in the second derivative of the curve [7] [42]. This method limits the flexibility of the estimated curve and prevents over-fitting. This

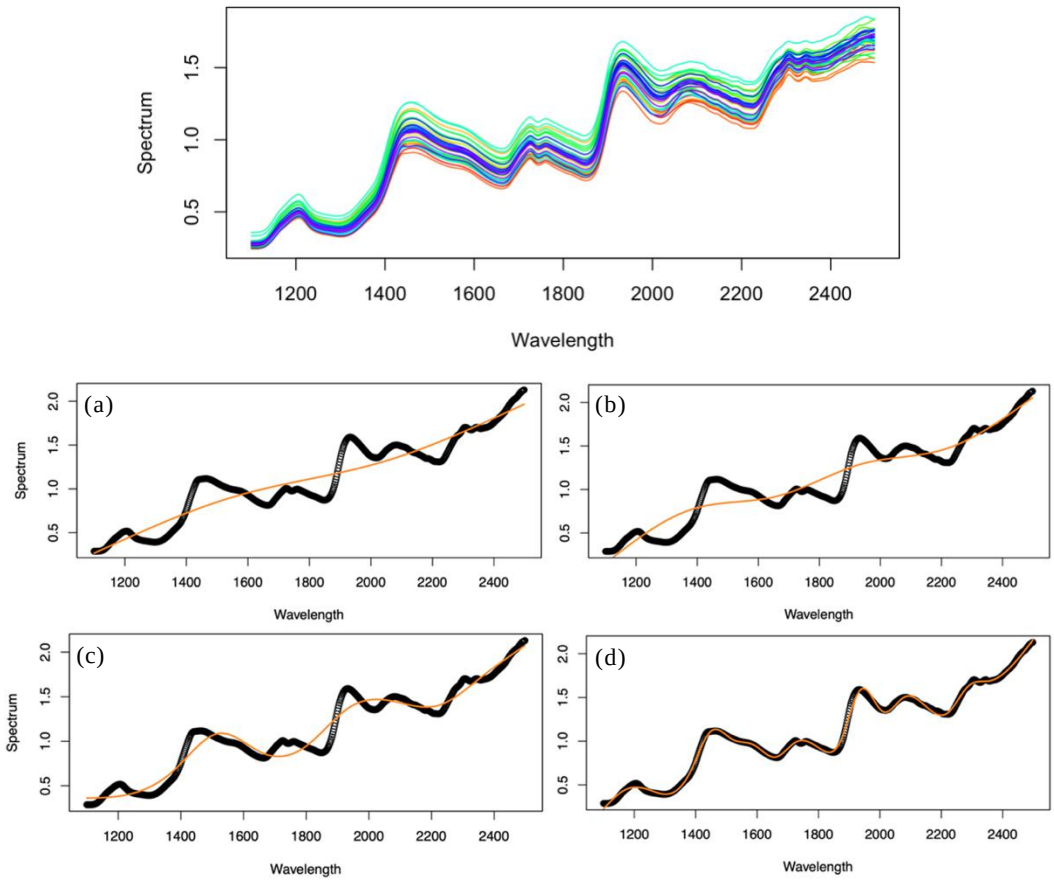


Figure 3.8 – Natural Cubic Splines on Biscuit Data *[34] . Biscuit data is one of the datasets available in fds package. (a) shows the Natural Cubic Spline with df=3. (b) shows Natural Cubic Spline with df=5. (c) shows Natural Cubic Spline basis df=10. (d) shows Natural Cubic Spline with df=20. As the degrees of freedom increases, the spline fit starts to capture the local fluctionas in the data. It imposes a trade-off between accuracy, smoothness and computational.*

type of spline utilizes a large number of equidistant knots and a penalty, based on coefficient discrepancies between successive B-splines polynomials [52]. As oppose to regression splines in which the number of knots controls the flexibility, it is the penalty component that controls the complexity in smoothing splines [51].Suppose that we intend to estimate *f(x)* by minimizing the following penalized residual sum of squares:

$$RSS(f,\lambda) = \sum_{i=1}^{n} \{Y_i - f(X_i)\}^2 + \lambda \int [f''(X)] d_X$$

Where $f''(\cdot)$ is the second derivative. The first term in the formula controls closeness to the real functional data points, while the second term imposes penalty on the curvature in the function. $\lambda \in (0, \infty)$ is referred to as the

λ	Curve	Implication
0	Very Rough	*f(X)* can be any function which interpolates between the data
i = 1, ..., n	Smooth	The solution is a cubic spline with knots at the unique values of X_i
∞	Very Smooth	The simple least squares fit, since no 2nd derivative can be tolerated

Table 3.1 – Effect of Different Values of Smoothing Parameters. *The smoothing parameters is included in the penalty component in Smoothing splines and controls the level of smoothness of the final curve.*

Smoothing Parameter that regulates complexity by providing a trade-off between two the terms involved in the penalty equation. To some extent, smoothing splines solve the knot selection problem. The strategy here is to employ a big number of equidistant knots and then allow λ to control the level of smoothness. However, there is no general rule on how to find the optimal value of the smoothing parameter. Some strategies are based on finding λ using generalized cross validation which we will discuss in the next section [7, 42].

3.6.5.2 P-splines

Smoothing splines use a large number of equidistant knots to fit the polynomial functions between them. But how many knots would be considered as a large number? In fact, If the number of knots is big enough, it is unnecessary to place a knot at every x_i where $i = 1,...,n$. This would lead to overfitting and high computational complexity. Instead, *Penalized Regression Spline* with a smaller number of knots can be used to approximate the smoothing spline. Penalized Regression Splines are the most often used types of Splines which implements the cubic B-spline basis functions on a huge set of equally spaced knots [15, 20, 51].

The degree of the splines, as well as the number of knots and their location, are included as the components of the smoothing parameters for regression splines. On the other hand, in Smoothing Splines the knots are already located on the *X-space*. In addition, cubic degree is usually utilized in smoothing splines. Therefore, the only parameter that must be tunned in Smoothing Splines is the penalty parameter λ. this leaves us with adjusting only one parameter for the P-splines which has added to its popularity and

ease of use. If certain simplifications are not implemented in regression splines, the process of selecting the location and number of knots for regression splines may be a hard task in terms of combinatorics [51].

In the future sections, we will go through the knot selection for regression splines in more details.

3.6.5.3 Selecting Smoothing Parameter

When applying penalized smoothing, the function of the smoothing parameter is to determine the degree to which the fitted curve is smooth. The two commonly used selection criteria are called *Leave-one-out Cross Validation (CV)* and *Generalized Cross Validation (GCV)* [7, 20, 51]. There are subtle differences between the two proposed methods that we refer to [17] for further information. In summary, It has been shown that The CV method has theoretically two primary flaws: first, it has a high computational cost; second, it has the potential to result in the data being under-smoothed. What is done throughout the CV method is that the N data points are divided into K groups, also known as folds. Then the model is trained on the remaining data while using each fold as a test set in turn. The prediction error may be assessed on the unseen K^{th} fold following model fitting with a given smoothing value. Each data point is tested once after this process is carried out for all folds, allowing the prediction error for all folds to be averaged. For a each smoothing parameter, the average prediction error offers an estimate of the test error of the curve. The smoothening parameter that minimizes the test error of the model is then chosen [7, 51, 53]. The GCV approach is less complicated to compute and has a long history of use in scientific research on smoothing splines [7, 18]. GCV method also has two versions depending on whether smoothing or P-splines are used. However, these statements might not be the case when it comes to practical applications. In most cases, none of these method mentioned above performed consistently better than the others, according to a comparison study [52, 53, 54]. Therefore, it is recommended to perform trial and error in order to make sure that the smoothness of the final curve has reached to its optimum level. It is worth mentioning that the incorporation of a smoothing penalty necessitates a modification in the fitting process in order to make room for penalty component. That is, the

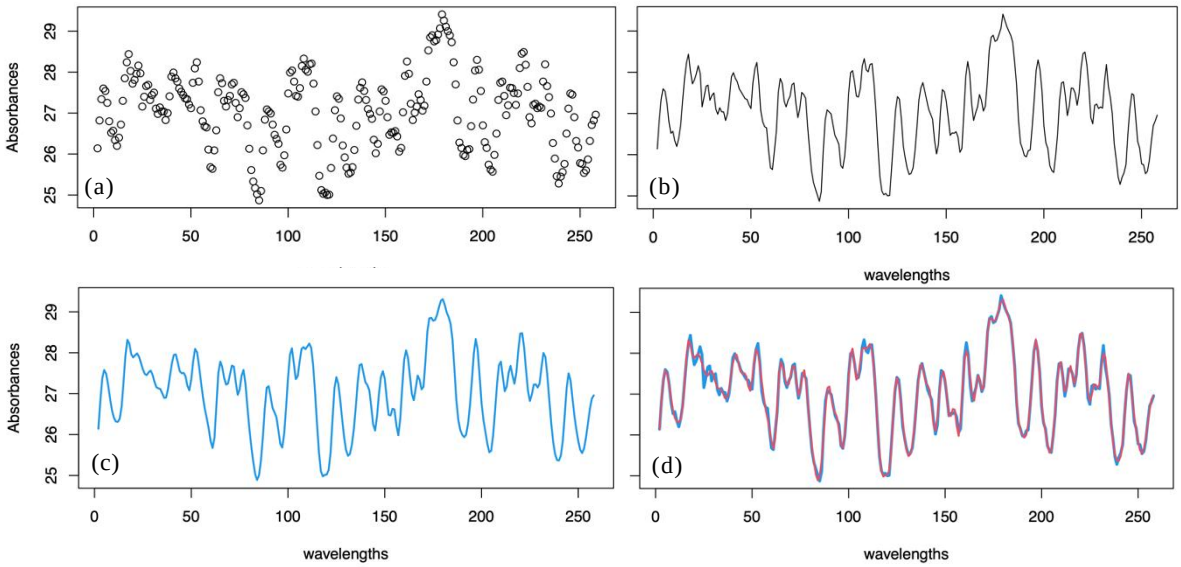


Figure 3.9 – Comparison of Smoothing Splines with P-splines on El-Nino. *The El-Nino data can be found in [36]. (a) shows the raw discrete data points. (b) shows the linear interpolation. (c) shows the smoothing spline fit which place a knot at every single point in the domain with CV. (d) compares the smoothing spline with P-splines. As the red line indicates, there are some small weakness in capturing local behavior of the curve.*

penalty component has to be included into each regression function between all the successive splines joining at knots.

3.6.5.4 Smoothing Splines in R

Splines package in R provides powerful tools to fit smoothing splines on data. The *smooth.spline()* fits a smoothing or panelized spline depending on the input arguments. There are many arguments that can be passed to the function as an input. However, the advantage of using this function is that it provides an interface and automatically performs the tuning parameters process and generates a smooth function for the given data. We will give an example using El-Nino time series available at [36]. Figure (3.9) illustrates the data. Complete code of applying smoothing splines on El-Nino data can be found in appendix (5):

As Figure (3.9) demonstrates, the smoothing spline which place equidistant knots, the blue curve has fully captured even the local fluctional of the data. The red curve is the P-spline that utilize a relatively smaller number of knots. Table (3.3) compares the smoothing parameters of these methods. Apparently, the complexity of Smoothing Splines is of higher orders compared to P-splines based on the degrees of freedom involved in two

methods. Table (3.2) reports the smoothing parameters for El-Nino using both CV and GCV method. It is worth mentioning that the incorporated penalty component in smoothing and panelized splines must be included in each of the adjacent spline functions meeting at the common knots. This will add to the complexity and imposes a trade-off between the power and the cost of the method should be taken into account. It might be one of the reasons why this method has been less used in practical applications even though it provides a sufficient amount of accuracy [20].

3.7 Placement of Knots in Regression Splines

The great level of flexibility that spline modeling offers comes at the cost of the number of tuning parameters, Basis functions, number and the location of knots. It turns out that the choice of basis functions and the degree of the underlying polynomials do not have nearly as much of an effect as the other two of these factors [20, 55]. In point of fact, spline fits are very robust to the degree of the polynomials. Cubic polynomials, with a degree of three, are the standard choice since the curves produced by cubic curves appear flawlessly smooth to human eye. If the derivatives of the fitted curves are of interest, then it may be useful to use a higher degrees. Although in general, fits with degrees greater than 3 are functionally unrecognizable from one another.

Number of knots, spacing, and whether or not to utilize a penalty function, such as the integral second derivative of the spline, are two of the most important decisions. As discussed in the previous section, the degree to which the resultant non-linear function is flexible is directly proportional to the total number of knots in regression splines.

Many have investigated the methods through which the an appropriate set of knots is selected. The choice of knots is an important problem when working with B-splines. If too many knots are selected you have an overfitting of the data. On the other hand too few knots provides an underfitting. This fact is specially significant in the case of non-penalized spline regression (regression splines). Some automatic numerical schemes for optimizing the number and the position of the knots were proposed to solve this problem [55].

Method	*spar*	λ	*Equivalent DF*	*Score*	*Time Taken*
CV	0.05417272	3.995e-09	100.765	0.02671309	0.005592
GCV	0.04409647	3.382e-09	101.325	0.02820579	0.005337

Table 3.2 – Effect of CV and GCV in Smoothing Splines*. Comparing the parameters of ordinary cross validation (cv) and generalized cross-validation (gcv) on El-Nino dataset [36].*

Method	*spar*	λ	*Equivalent DF*	*Score*	*RSS*	*Time Taken*
Smoothing Spline	-0.8273256	4.597e-17	257	0.01140677	3.511e-16	0.0117321
Panelized Spline	0.04409647	3.382e-09	101.324	0.02820579	2.659774	0.01195407

Table 3.3 – Parameters of Smoothing Splines and P-spline*. Comparing the parameters of Smoothing parameters of Smoothing splines and P-splines on El-Nino dataset [36].*

In Regression Splines, the choice of number of knots and their location significantly contributes to the level of smoothness of the spline curve. By convention equidistant knots are selected. Note that the smoothing also use dense equidistant knots to fit the polynomial splines between them. However, depending on the input data, other methods has been proposed to be adopted in order to obtain the most efficient application-specific knot vector. Paul and Brian in their study compared B-spline using equidistant knots with truncated power series splines where knots are based on quantiles of the independent variable [15]. The conclusion indicated that the B-spline with equidistant knots is preferred. Klein also provides an overview of the applicable knot selection strategies including, equidistant, residual and Quantile in his paper [56]. It was observed that the equidistant knots could make the spline sensitive to the data outliers and may result in lower accuracy compared to other methods.

The choice of knots also has important implications for sparse functional data. When data records are not present in some intervals, it becomes difficult for the model to capture the behavior of the data for that particular range. Therefore, by placing a set of knots in the sparse region, it is not guaranteed that that the model performs accurately. Using a method to adaptively select the number and the locations of knots is as an advantage to tackle this problem. A new method has been proposed through which the knots are selected specific to the input data characteristics [16]. In nutshell, they have trained a model which predicts the number and the location of

appropriate knots. The implementation of this technique is available in R package called DDK (Data-Driven knots). The experiment was conducted on the a functional data set called Wine, described in the original paper [16]. The result of the experiment showed a significant decrease in Mean Square Error of the final fit compared to choosing equidistant knots. We will examine this method on a set of different functional dataset to evaluate the implications in the next sections.

3.7.1 Equidistant vs Quantile Knot Placement

The term uniform and nonuniform are interchangeably used for equidistant and irregular knot placement respectively. One common methods of knot placement is known as *Quantile Knot Sequence*. In this method of knot placement, the quantiles from the empirical distribution of the underlying data are used as the interior knots. The use of Quantile Knots ensures that the same number of sample observations will be located in each interval, despite the fact that the intervals will be of varying lengths [13].

	MSE	RMSE
Equidistant	0.1480091	0.3847195
Quantile	0.1549008	0.3935744

***Table 3.4** – Comparison of Quantile and Equidistant Knot Placement*

For instance, Suppose we have a functional data with estimation function *f(x)* defined by the formula below. To fit the spline using 20 knots:

$$f(x) = -\sin(x) + 0.2x + \cos(|x|)$$

Figure () visualizes the data in the range [-10, 10]. We can either place the knots on the 1st to 20th Quantiles of the data range or we can place them uniformly based on equal distance. Figure () and () visually compares two methods proposed. Table (3.4) also reports the MSE, RMSE of the fitted curve compared to the original functional data.

As briefly mentioned earlier, Quantile Knots place equal number of data samples in each interval as opposed to equidistant knots in which different number of knots lie between the intervals. Therefore, the Equidistant Knot Placement would be a better choice if data contain noise or outliers as the

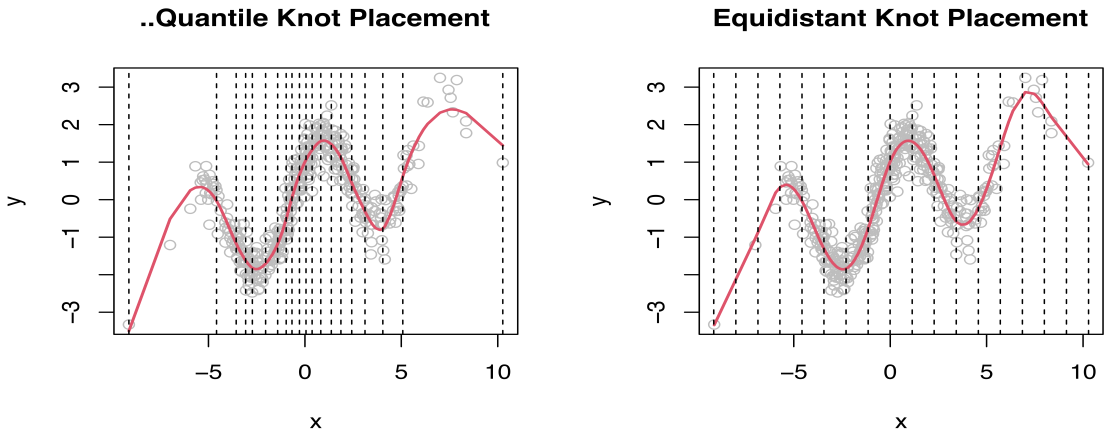


Figure 3.10 – Equidistant and Quantile Knot Placement *The figure above shows the raw discrete values of data. The left figure shows the result of quantile knot placement using cubic B-spline and the right shows the results of equidistant knot placement. In this function, the equidistant method outperforms the quantile as it has captured the local behavior of the data.*

effect of the outlier will be mitigated and the final curve appears more even. However, As Table (3.4) indicates, the values of the errors are quite similar and it is recommend to apply both methods and observe the results before making the final choice. The results are in line with the finding of Paul and Brian in which the combination of equidistant knots and B-splines are the best applicable choice for a wide variety of datasets [15]. The complete code for comparing Quantile and Equidistant knot placement can be found in appendix (6).

3.7.2 Data-driven knot placement

In this section, we will conduct an experiment on a set functional datasets to examine the Data-driven Knot Placement [16] for selecting optimal knot vector for spline fitting. As briefly discussed earlier, the core idea of this method is based on training a model which can predict the optimal number of knots and their location based on the main input features of the data. The placement of knots plays a key role in regression splines. Although in most cases equidistant knot placement is preferred and works relatively accurate, the result could be different in some sort of functional datasets. Finding proper knots vector would involve performing some trial and error. However, if we utilize a model which helps us to find the most efficient knot placement, we can simply rely on the prediction of the trained model. Depending the characteristic of the given functional data, the placement of knots would differ. Therefore, it would be beneficial if we predict the knots placement based on the features extracted from the input data. That is why this

method called Data-driven Knot. The approach makes advantage of the positioning of the knots and related spline bases to build a low-dimensional method that is resilient with regard to the kind of data that is being used [16] . While the proposed method is based on a mathematical analysis, we focus on the practical use of the Data-driven knot placement and refer to the original paper for more details [16]. We will use the DDK package in R [57] to apply the method on two types of functional datasets to understand how it works and what implication would it have.

3.7.2.1 Moisture Data

This data set consists of near-infrared reflectance spectra of 100 wheat samples, measured in 2nm intervals from 1100 to 2500nm, and an associated response variables, the samples' moisture content [34] . As Figure (3.11 b) illustrates, the model has provided the location of 20 knots on the X-space. Which are:

$\xi^* = (1, 24, 33, 61, 105, 114, 135, 144, 151, 214, 239, 288, 374, 382, 411, 453, 545, 556, 621, 636, 648, 665)$

Figure(3.11(c)) reports the training and test errors of the models. Figure (3.11 (d)) visually compares the result of the two approaches. As it can be seen, in this case the spline with equidistant knots has outperformed the data-driven approach. As it has fully matched the original data. The moisture data however has been densely sampled and the nature of the data is already smooth. To generalize the finding of this section we carry out the same process on a noisier data set in the next section. The code can be found in appendix (7).

3.7.2.2 Phoneme Data

We extend the experiment to another data set called Phoneme which can be found in [37] . We first use the data-driven model to predict 10 knots. As illustrated in Figure(), the knot vector given by the model is as follows:

$$\xi^* = (0,8,42,57,69,77,96,108,120,149)$$

Figure (3.12 b) compares the visual result of the two methods. As it can be seen, this time, the data-driven method outperforms the equidistant model

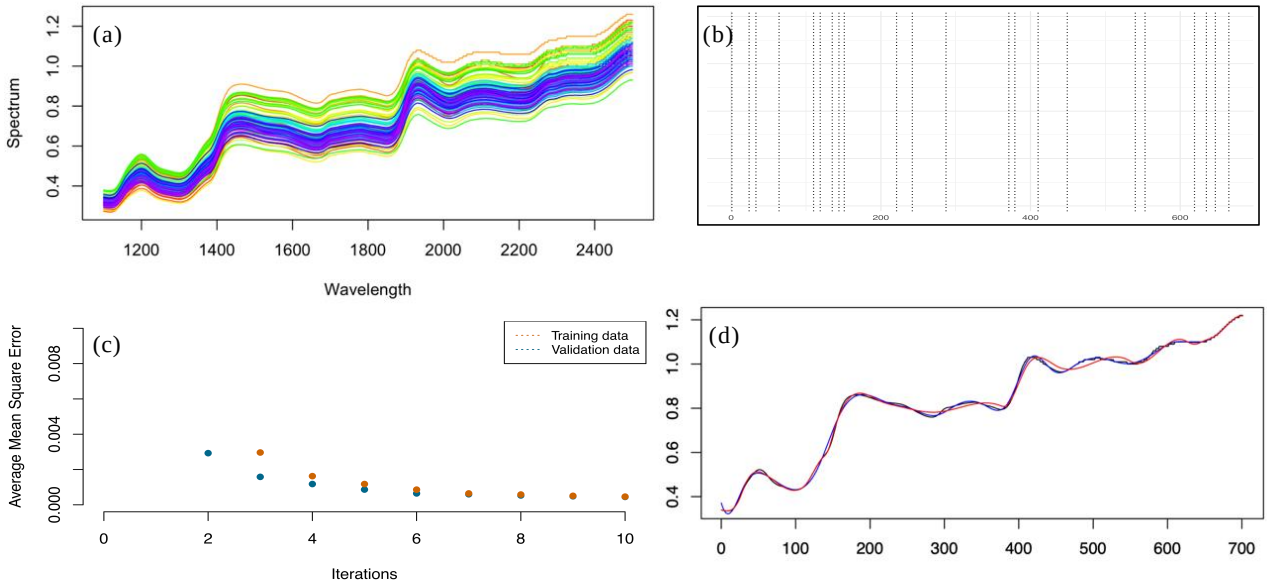
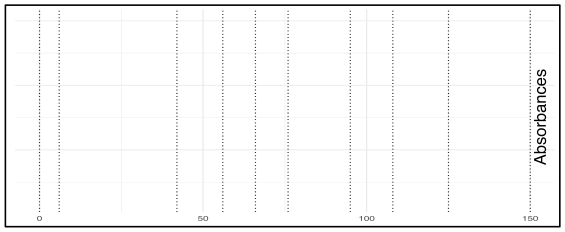


Figure 3.11 – Data-driven Knot Placemat on Moisture data. *Moisture dataset is a built-in dataset in fds package. (a) shows the whole functional dataset. (b) denotes the position of knots predicted by the model. (c) is the training and test average mean squared error. (d) compares the data-driven spline (red line) and the equidistant spline (blue line). In this case, the equidistant spline outperforms the DDK method.*

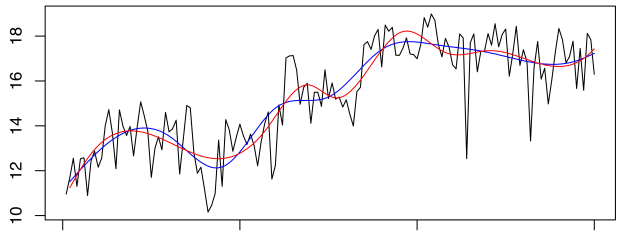
and it has captured the pattern of the data more accurately. Table (3.5) supports this fact. To generalize this notion, we augment the knot vector and use the model to obtain 20 data-driven knots. The knot vector generated by the DDK model is as follows:

$$\xi^* = (0,6,10,31,37,42,50,56,61,66,70,76,81,85,91,95,101,108,117,126,135,149)$$

Figure(3.12 (e)) depicts the result of the experiment. It can be seen, the data-driven has produced more accurate results and it has more control over the local fluctuations in the data. To conclude, we can say that based on the nature of the input data, the choice of knot placement strategy would vary. Having considered the results yielded by two different strategies on two different data set, we can deduce that if data sampling is dense and we the original curve is relatively smooth, we can simply employ a set of default equidistant knots and avoid the overhead of training model. On the other hand, if data is relatively sparse and the original curve appears jagged, the data-driven knot placement yields more accurate results and it is worth relying on the model. As it can be inferred from the visual results, this method outperforms the equidistant knots and the it has almost fully captured the pattern of the original function. However, it should be noted

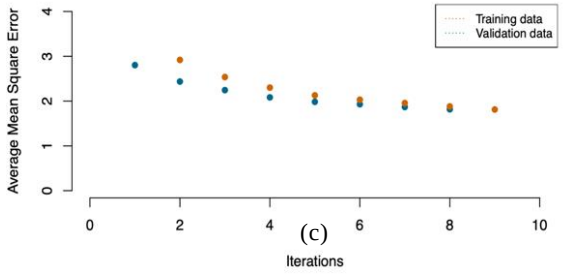


(a)



(b)

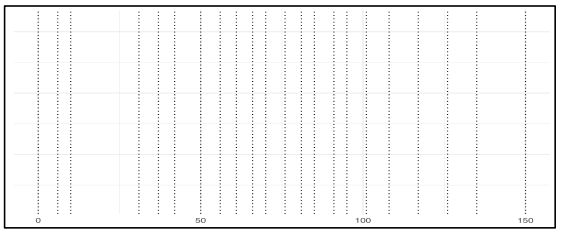
wavelengths



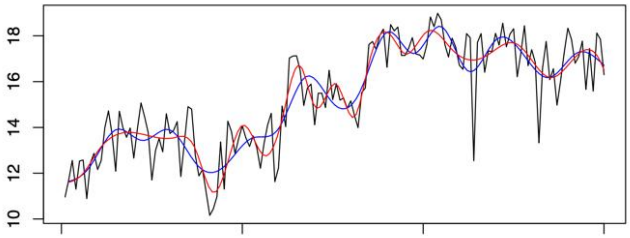
(c)

Method	RMSE	Time Taken
Equidistant	1.100	0.01134
Data-driven	1.077	0.01777

***Table 3.5** – Comparison of Data-driven and Equidistant Knot Placement with 10 knots,*

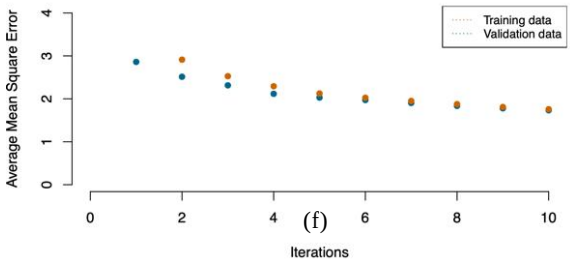


(d)



(e)

wavelengths



(f)

Method	RMSE	Time Taken
Equidistant	0.982	0.00712
Data-driven	0.916	0.00646

***Table 3.6** – Comparison of Data-driven and Equidistant Knot Placement with 20 knots,*

***Figure 3.12 – Data-driven Knot Placemat on Phenome data.** Phenome dataset is a built-in dataset can be found in [37]. (a) shows denotes the position of 10 knots predicted by the model. (b) compares the data-driven spline (red line) and the equidistant spline (blue line). (c) is the training and test average mean squared error. (d), (e) and (f) show the same things for 20 knots. The Data-driven knot placement outperforms the equidistant method.*

that this method adds an overhead as it requires the model to be trained to determine the number and the location of the knots. We then construct the spline based on the information provided. However, it is an example of how the knot selection impacts the final curve and to what extent it can add to the level of the smoothness. The code can be found in appendix (7) and (8).

Chapter 4

Wavelets

Wavelets can be considered as a next generation of the Fourier transforms [51, 23]. To understand wavelets, it is required to see how Fourier transforms work. Fourier Transform are used to transform a signal from its time-domain to its frequency-domain. In other words, Fourier Transform indicates what frequencies are present in the curve. However, it only has resolution in frequency-domain, meaning that, it is not capable of providing any information about the location of where the frequency happens. When the frequency spectrum is steady, Fourier Transforms perform well. However, the more dynamics is added to the curve , the more difficult it becomes to locate the frequency. This is not desirable since most of the data we see in real world has dynamic frequency. Wavelet Transform have been introduced to address the limitations of Fourier Transform.

Suppose there is a periodi data, described by the formula below:

$$f(x) = \sin(2\pi.10x) + \sin(2\pi.50x) + \sin(2\pi.200x) + \sin(2\pi.300x) \quad (4.1)$$

From the formula, we can deduce that the frequencies present in the curve are 10^{Hz}, 50^{Hz}, 200^{Hz} and 300^{Hz}. (See fFgure (4.1(a))). If the curve is tranformed into its frequency-domain using Fourier transform, it shows the frequencies respectively. (Figure 4.1(b)) Suppose another function that comprises of the concatenation of the functions in formula below:

$$f(x) = \begin{cases} \sin(2\pi.10x) & 0 < x < 0.25 \\ \sin(2\pi.50x) & 0.25 < x < 0.5 \\ \sin(2\pi.200x) & 0.5 < x < 0.75 \\ \sin(2\pi.300x) & 0.75 < x < 1 \end{cases} \quad (4.2)$$

The first term is present in the first quarter of the curve, the second one term the second quarter and so on. It is crucial to observe that the two frequency spectra have the same peaks, therefore we cannot distinguish where these frequencies are present in the signal. That is why the Fourier Transform cannot tell the difference between the first two signals.

An enhanced version of Fourier Transform called Short-Time Fourier Transform were also introduced to address this limitation. The idea is before performing the Fourier Transform, the original curve is segmented into equal-length portions, which may or may not overlap. Then the transform in performed by sliding through the curve using a window. For instance, suppose the data has been split into five segments, If the Fourier transform detects a certain frequency in the second segment, it can be claimed that the frequency takes place in the second segment and we can locate the Frequency. The problem with the proposed method is that the smaller segments are made, the more accurately we can locate the frequency but the less we know about the amplitude of the frequency itself. On the other hand, if we make bigger segments, the less accurately we can locate the frequency. Wavelet Transform overcomes the problems mentioned above. It has high resolution both in time- and frequency-domain. That is, not only it does provide information about the value of the frequency, but also it gives information about the time and location where this frequency takes place.

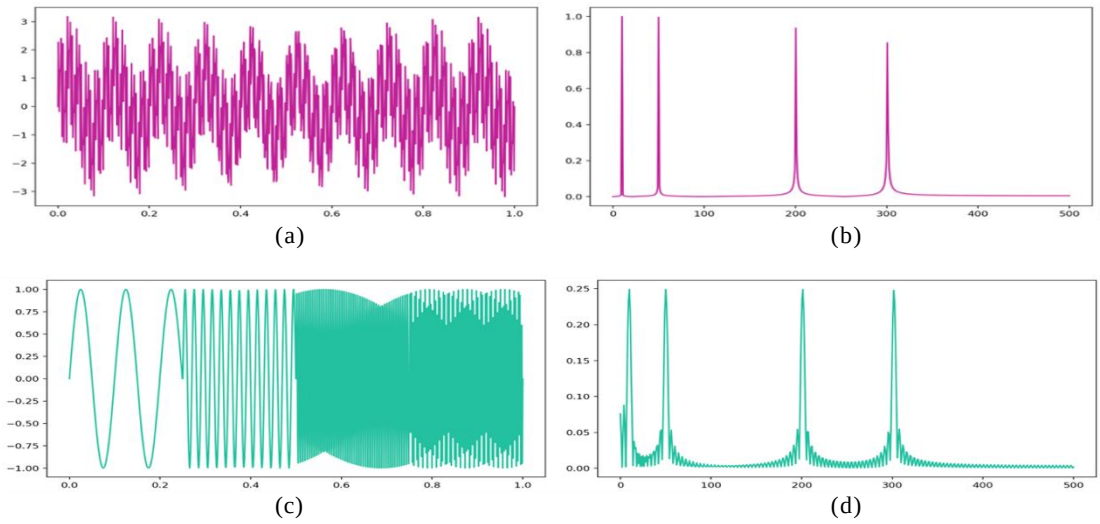


Figure 4.1 –Frequency Resolution in Wavelets *(a) shows the the wave generated by Eq(4.1). (b) shows the wave in its frequency-domain. (c) shows the the wave generated by Eq(4.2) and (d) shows the wave in its frequency domain. (b) and (d) show the same frequency despite the their time-domain equivalent being different, indicating the lack of having time resolution in Fourier Transforms.*

This versatility of the wavelets are due to the fact that they use different scales. More generally, there is a trade-off in wavelet transforms. At scales where time-dependent aspects are more of interest, it has high resolution in time-domain. Likewise, at scales where frequency-dependent aspects becomes more important, it will have high resolution in frequency-domain.

4.1 How Wavelets Work

In Fourier transform, the goal is to represent the function using a series of sine and cosine terms, resulting in a linear combination of sine-waves. The Wavelet Transform instead uses a series of function called Mother Wavelets with different scales.

As we can see in Figure (4.2), unlike sine wave which stretches out from throughout the domin, the wavelet function is localized in time. This is the advantage of the wavelet which allows obtaining time-dependent information from the data. Since Wavelet functions are localized in time, we can multiply an input function with the Wavelet at various points in time. The multiplication starts from the beginning of our curve and gradually shift the wavelet to the finishing point. This process is also referred to as a Convolution [25]. After we have done this for the original Mother Wavelet, we scale it up and repeat the procedure. The basic formula of wavelets is :

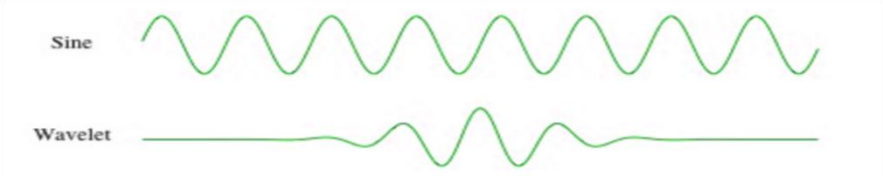


Figure 4.2 –Comparison of Fourier Basis and Wavelet Basis *The sine wave which form the Fourier transform basis stretches out the whole domain whereas a wavelet basis is localized in time, enabling wavelets to have high resolution both in time- and frequency-domain [25].*

$$X_{a,\,b} = \int_{-\infty}^{\infty} x(t)\psi_{a,b}(t)dt \quad (4.3)$$

Where x denotes the original function, ψ is a mother wavelet, a denotes the scale, and b denotes the translation factor. $X_{a,b}$ is the Transformed function.

4.1.1 Wavelet Scale

The functionality of the scale factor in wavelets is the same as the window size described in the previous section. Because the term *Frequency* has been previously used for the Fourier Transform, the wavelet transform is often described in scales rather than frequencies. However, there is an equation through which we can obtain the frequency from the scale and vice versa [25].

$$f_a = \frac{f_c}{a} \quad (4.4)$$

where f_a denotes the pseudo-frequency, f_c indicates the frequency present in the center of the mother wavelet and a is the scaling factor. As the formula suggests, larger scale-factors (longer wavelet) result in smaller frequencies. Therefore, by scaling up the wavelet in the time domain, we study smaller frequencies and achieve greater resolution in the frequency domain. Similarly, by employing a smaller scale, we may see more information in the temporal domain. Scales are essentially the inverse of frequency.

4.2 Different Wavelets

Wavelets are classified into several distinct families. The wavelet families differ because each family has chosen a distinct trade-off in how compact and smooth the wavelet appears. This implies we may select a wavelet family that best matches the characteristics of the inputdata. Each form of wavelets

has a unique shape, smoothness, and compactness that makes it useful for a certain application. Because a wavelet must meet only two mathematical constraints, it is simple to create a new form of wavelet. These two constraints are called normalization and orthogonalization. That is, a wavelet must have

1) Finite Energy:

A wavelet must be localized in time and frequency. It is integrable and the inner product between the wavelet and the original function always exists.

2) *Zero Mean:*

A wavelet has zero mean in the time domain. This is required to verify that it is integrable and that the inverse of the wavelet tan transform can be computed.

Within any wavelet family, there may be a number of distinct wavelet subclasses. The number of coefficients, also referred to as the number of vanishing points, and the amount of decomposition define the various wavelet subclasses. The number of folding present in the mother wavelet function can vary. This introduces the notion of *Vanishing Moments*. For instance, db3 has three vanishing moments while db5 has five. The number of vanishing moments is proportional to the wavelet's approximation order and smoothness. With *d* vanishing moments, a wavelet may approximate polynomials of degree *d-1*. Figure (4.4) illustrates that as the number of vanishing moments grows, so does the wavelet's polynomial degree and it gets smoother.

4.3 Continuous and Discrete Wavelet Transform

As described before, the Wavelet Transforms encompasses two distinct categories, Continuous Wavelet Transforms and Discrete Wavelet Transforms. Suppse *a* and *b* are the scale and translating factors of a wavelet respectuively. In Continues Wavelets, These factors can take any continuous values. For instance, a continues mother wavelet can be scaled or translated by a factor of:

$$a, b = \{0.1, 0.11, 0.111, 0.111, \dots\} \quad (4.5)$$

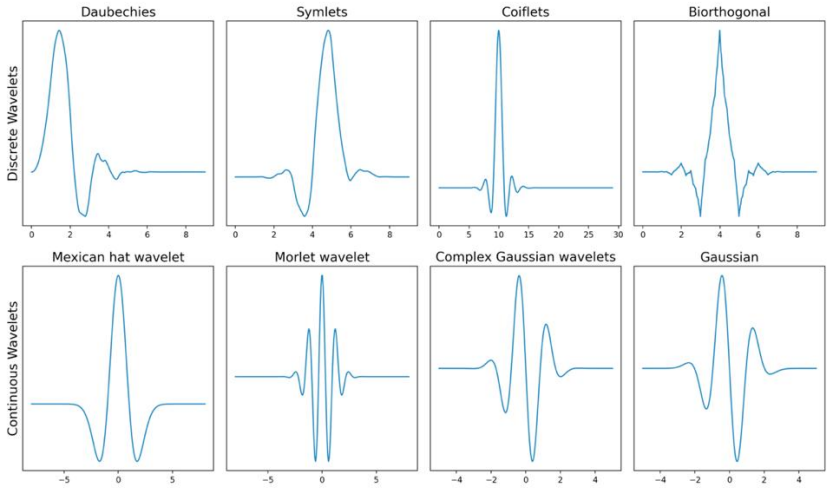


Figure 4.3 –Different Mother Wavelets *These figures show the mother wavelets available in PyWavelet library. Continues Wavelet transform and Discrete Wavelet Transform are the two categories of wavelets. The mother wavelet is scaled and translated and will convolved to the original function in the transformation process. The shape of mother wavelet has a major impact on how*

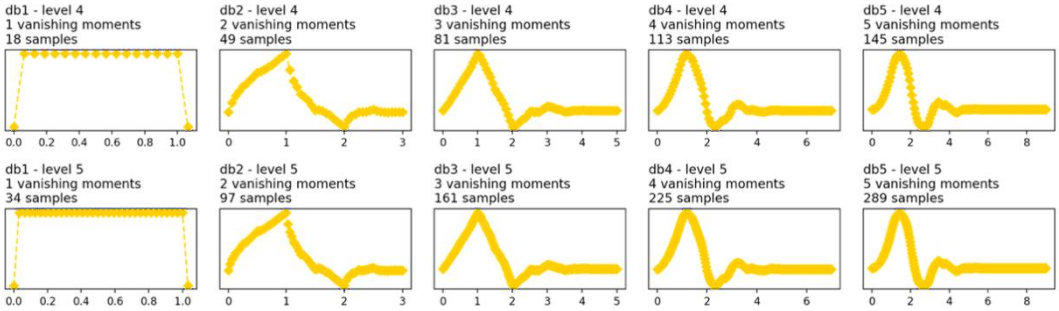


Figure 4.4 –Vanishing Points in Wavelets *As the number of vanishing point grows, the level of smoothness of the wavelet increases which results in having smoother output curves.*

The fundamental difference between the Discrete Wavelet Transform and the Ordinary Wavelet Transform is that the DWT employs discrete values for the scale and translation factor. So that in Discrete Wavelets the the scale factor rises in power of two:

$$a = \{1, 2, 4, 8, \dots\} \qquad (4.6)$$

Moreover, the translation factor raises integer values.

$$b = \{0, 1, 2, 3, 4, \dots\} \qquad (4.7)$$

4.5 Smoothing with Discrete Wavelet Transform

Since the main purpose of this paper is to provide smoothing tools, it is time to see how Wavelets are employed in smoothing context.

Apparently, presence of high frequency data makes the curve appear noisy and jagged. A smooth curve can be made by removing/detaching the high frequency from the data. This method can be practiced by defining a threshold and removing the frequencies higher than the defined threshold [25, 58]. The DWT is considered as a tool to break down the data into a number of frequency subbands by implementing a series of of high- and low-pass filters known as *Filter-banks*. The resultant of the high- and low-pass filters are called *Detail Coefficient* and *Approximation Coefficitens* respectively. The approximation coefficients are the one which is used to creconstruct the smooth approximation of the curve. Many have used Wavelet Transforms as a tool to apply smoothness to a set of application-specific data [59] [30].

As mentioned earlier, the process of applying Wavelet Transform to the curve begins by multiplying a mother wavelet with the smallest scale to the data. According to Eq (4.4), small scales correspond to high frequencies. This implies that we examine high frequency behavior first. Since we are working with DWT, the scaling factor utilizes discrete values e.g. 1, 2, 4, 8. When the scale grows by a factor of two, the frequency inveresly lowers by a factor of two. Hence, in and around half of the maximum frequency. The same process applies in the third-level when the scale increases by a factor of four. We securitize the data at around one-fourth of the maximum frequency and so on. This process continues until we arrive at a state which is referred to maximum level of decomposition.

When the data has been decomposed into a set of frequency subbands, it is clear that the number of samples in the data has also been reduced by a factor of two at each succeeding level of decomposition (As a result of performing downsampling to remove high-frequency samples). The maximum level of decomposition is then defined as the level in which the number of samples in the data are lower than the wavelet‘s.

4.6 Choosing a Mother Wavelet

So far, we have discussed the fundamental concepts of the Wavelet Transform. What is clear is that based on the choice for mother wavelets, the output of the transform is completely different. Therefore, the choice of mother wavelet becomes of crucial importance. Some have investigated the

methods through which a suitable mother wavelet can be chosen for a given dataset and discusses the optimal solutions to select the suitable mother wavelet and the level of decomposition to denoise the data [26, 27, 60]. It has been found out that the length of the mother wavelet is the first criteria that should be taken into account. It is also suggested that the level of decomposition should be chosen proportional to the sampling frequency of the dataset. The higher the sampling frequency is, the more effective the denoising process will be. There has also been some intelligent data-driven based method proposed for choosing optimal mother wavelet [28] . A good review of the wavelet experiments and their justification for choosing the relevant mother wavelet can be found in . It suggests that the shape of the mother wavelet is what that determines the final form of the transform [44].

Finding suitable mother wavelet for any given dataset is still an ongoing field of study [44]. Most of the proposed methods have been based on trial and error indicating that it is still an inventible component of the analysis. However, since we are looking for smoothing techniques, we have intuitive criteria to begin with. The two main factors that significantly impact the shape of the curves are as follows:

1) Shape of the mother wavelets
2) Number of vanishing points

The smoother the shape of the mother wavelet is chosen, the softer the final curve becomes. If we chose a mother wavelet which is made of straight acute lines, we cannot expect the final curve to be smooth. The implications of the smoothness of the mother wavelet is similar to the degree of the polynomial in splines. Number of vanishing points also works is the equvalent of the degrees of freedom in splines. Although higher number of vanishing points add to the local control of the curve, however, it imposes a trade-off, high number of vanishing points can make the curve capture the local noises and contently reduce the smoothness of the curve.

We will apply denoising using DWT on a set of functional data to fully grasp the notion. Consider the Phoneme data illustrated in Figure (4.5 (a)). The description of the data can be found in [37]. Figure (4.5 (e)) depicts the results of low- and high- pass filters up to five level of decompostion. As the level of decompoition goes up, the approximation coefficents provide more

abtract representation of the curve. The right-side plots in Figure (4.5) show the wavelet transform of the data using *sym* mother wavelet using different values for vanishing points. As we are trying to produce smooth curves, the more vanishing points are employed, the smoother the outcome will be. It can be seen that the decomposition with sym20 also captures the local fluctuations of the original curve. Table (4.1) compares the Mean Squarred Error of the transforms and the time taken by each.

Figure(4.6) is the result of using different mother wavelets. The wavelet transform has been performed on the data using all available mother wavelts. We then applied a cross-validation to chose those with higher accuracy. The transforms shown in Figure (4.6) have been chosen out of all possible outputs and showed the the best results in terms of accuracy. Table (4.2) compares the results in terms of accuracy and time complexity. Although the accuracy has increased as we use wavelets with more vanishing points, however, the smoothness of the curve will be compromised. Sym16 seems to be a good compromise between the accuracy and the smoothness as the Table (4.1) and Figure(4.5) show. The complete code filter-bank is in appendix (10) for the wavlet on phoneme data can be found in appendix (9).

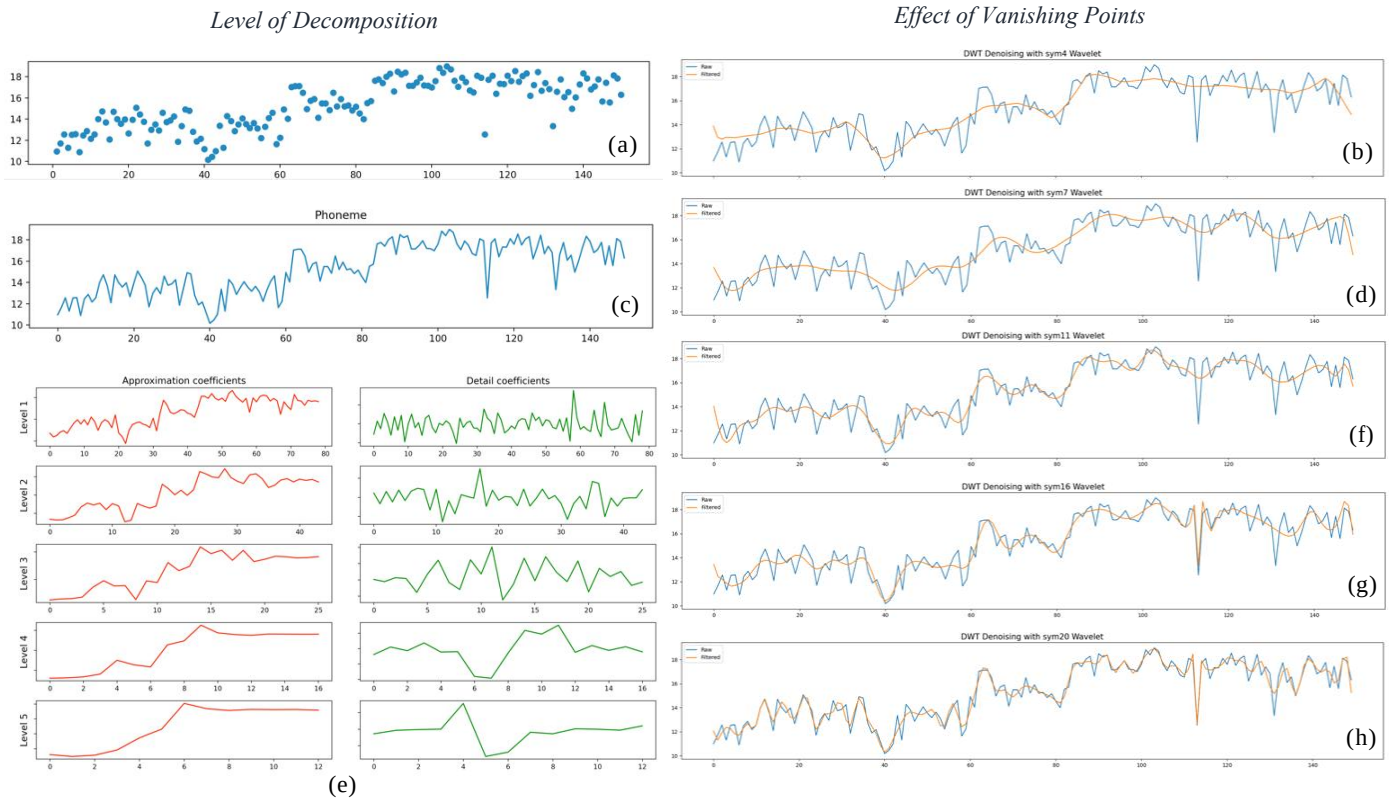


Figure 4.5 – Effect of Level of Decomposition and Vanishing Points *The left-side figures are to demonstrate the effect of different level of decomposition in Wavelets. (a) and (b) show the Phenome data which can be found in [37]. The approximation coefficients in (e) indicated by red lines are the results of the low-pass filters and the right-side figures indicated with green are the result of high-pass filters. Note that higher levels of decomposition result in more abstract representation of the data. Figure (b), (d), (f), (g) and (h) show the out of Wavelet Transform with one level of decomposition and with 4, 7, 11, 16 and 20 vanishing points respectively. The more vanishing points are employed, the more local behavior of the data is captured.*

Wavelet	*Vanishing Points*	*MSE*	*RMSE*
sym	*4*	*1.1422*	*1.0687*
sym	*7*	*1.1204*	*1.0585*
sym	*11*	*0.8256*	*0.9086*
sym	*16*	*0.7909*	*0.6256*
sym	*20*	*0.5912*	*0.3495*

Table 4.1 – *Evaluation metrics of using different vanishing points for mother wavelet on Phenome Data [37]*

Wavelet	*Vanishing Points*	*MSE*	*RMSE*
sym	*16*	*0.7909*	*0.6256*
db	*19*	*0.8097*	*0.8998*
coif	*6*	*0.8016*	*0.8953*
bior	*1.5*	*1.0649*	*1.0319*

Table 4.2 – *Evaluation metrics of using mother wavelets on Phenome Data [37].*

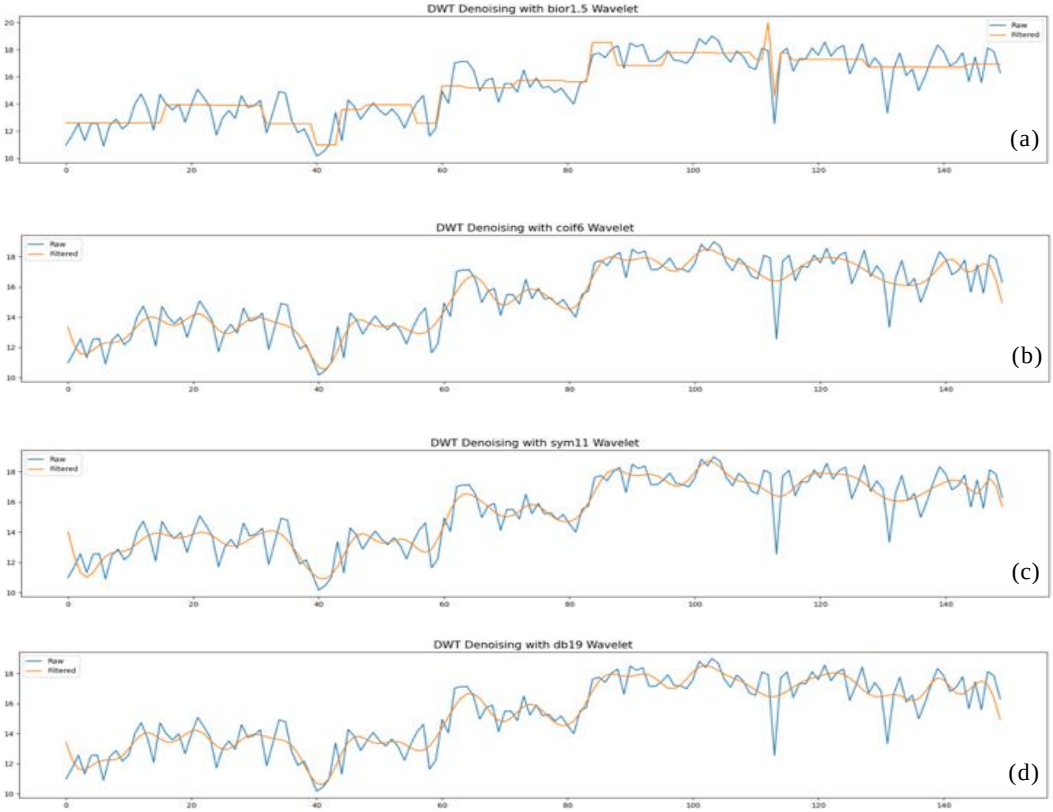


Figure 4.6 – Comparison of Different Wavelets on Real Data. *All the plots shown in this figure have been selected after applying cross-validation on different choices of mother wavelets. The shape of mother wavelets and the number of vanishing points employed in the transform are the key factors to make a smooth curve from the data. (a), (b), (c) and (d) are the resultant of applying bior 1.5, coif 6, sym 11 and db 19 mother wavelets respectively.*

Chapter 5

Practical Guide

The intention of this section is to provide a step-by-step tutorial through which some of the smoothing techniques discussed in this study can be applied on a set of raw functional data. We begin with a brief description of the data. Then we discuss the available packages for applying splines and wavelets. We then provide the code snippets alongside with the explanation. In the final secretion, the results are discussed.

5.1 Data Description

The experiment will be carried out on Dublin Bike Availability. The original data of Dublin Bike Stations can be found in [38] . The data consists of information of Dublin Bike stations including longitude, latitude, number of available bikes, number of bikes taken and etc. The data provides the information from Wednesday, January 1st 2020 through Sunday, March 22nd 2022. The sampling schedule is every 5 minutes.

5.2 Review of Spline Packages

The practical guide for splines has been implemented in R. R is one of the most popular program among research statisticians, or those who conduct statistical data analysis, is R. It has an open software license that and is free to use. R provide powerful tools to apply splines and these tools have been comprehensively reviewed by Aris his paper [20]. According to another study carried out by Aris, the spline packages in R comprise of two categories, Spline creators and regression packages [20, 61] . There are several spline creator packages available in R, some of which alongside with their description are listed below. Table (5.1) compares the useful spline creator packages available in R. Having vignettes, supplemental documentation e.g. website, book, practical guide, real life data and clear examples of functionalities are a set of criteria which defines a good package. Other packages are mostly based on the packages presented in Table(5.1) and implement similar functionalities. Few others have been designed to address application-specific requirements.

Table(5.2) shows an evaluation of different regression packages available in R from multiple aspects. While the The detailed description of each package is out of the scope of this chapter, An evaluation of these packages alongside with a brief description of each has been reported in Table (5.2). For more details, we refere to the Aris's papaer [20, 61]. For the purpose of this study, we implement the practical guide using splines package. The codes for previous sections are also implement in R using splines and mgcv packages which can be found in relevant appendix.

Package	*Description*	*Download*	*Reverse Dependency*	*manual*	*Installation guide*	*Visual Examples*
splines	General smoothing and regression spline	*Built-in*	*124*	✓	✓	✓
splinet	Splines and Orthogonal Spline Bases	*3k*	*0*	✓	✓	✓
gss	General smoothing and regression spline	*24k*	*5*	✓	✓	✗
polsplines	Polynomial spline functions	*21k*	*0*	✓	✓	✗
crs	Categorical Regression Spline functions	*19.8k*	*2*	✓	✓	✗
logspline	Log spline density estimation functions	*7.6k*	*3*	✓	✓	✗
pspline	Polynomial spline functions	*5.3k*	*6*	✓	✓	✗
MBA	Multilevel B-spline Approximation	*4.9k*	*6*	✓	✓	✗
cobs	Constrained B-Splines	*4.8k*	*2*	✓	✓	✗
bezier	Bezier Curve and Spline Toolkit	*2.4k*	*1*	✓	✓	✗
pbs	Periodic B-Splines	*1.1k*	*0*	✓	✓	✗
bigsplines	Smoothing Splines for Large Samples	*0.54k*	*1*	✓	✓	✗
Orthogonalsplinebasis	Orthogonal B-Spline Functions	*0.39K*	*0*	✓	✓	✗
Kpart	Spline fitting	*0.27*	*0*	✓	✓	✗
sspline	Smoothing Splines on the Sphere	*0.16k*	*0*	✓	✓	✗

Table 5.1– Comparison of spline creators packages in R. *The table compares the spline creator packages available in R. The number of downloads is on a monthly basis. Reveres Dependencies means how many other packages use the original package in their underlying implementations. Splines packages are reviewed alongside with some visual examples in [20].*

	Packages						
	mgcv	quantreg	survival	VGAM	gbm	gam	gamlss
Criteria	*General Features*						
Downloads	*81k*	*307k*	*103k*	*34k*	*39k*	*34k*	*11k*
Vignette	✓	✓	✓	✓	×	×	✓
Book	✓	✓	✓	✓	×	✓	✓
Website	×	×	×	✓	✓	✓	✓
Dataset	*2*	*7*	*33*	*50*	*3*	*1*	*29*
Criteria	*Regression Models*						
Linear	✓	✓	×	✓	✓	✓	✓
Non-linear	×	×	×	✓	×	✓	✓
Categorical	✓	×	×	✓	✓	✓	✓
Count Regression	✓	✓	×	✓	✓	✓	✓
Survival	✓	✓	✓	✓	✓	✓	✓
Quantile Regression	×	×	×	✓	✓	✓	✓
Multivariate	✓	×	×	✓	×	✓	✓
Reduced Rank	×	✓	×	✓	×	✓	×
Other	✓	✓	×	✓	✓	✓	✓
Criteria	*P-spline Support*						
P-spline	✓	✓	✓	✓	✓	✓	✓
Criteria	*Post-fit Functions*						
Retrieving Coefficients	✓	✓	✓	✓	✓	✓	✓
Retrieving Predictors	✓	✓	✓	✓	✓	✓	✓
Plot	✓	✓	✓	✓	✓	✓	✓

Table 5.2– Comparison of regression packages in R. *The table compares the regression packages available in R. The number of downloads is on a monthly basis. Reveres Dependencies means how many other packages use the original package in their underlying implementations. A detailed overview can be found in [20].*

5.3 Review of Wavelet Packages

There have been a few wavelet-related packages available both in R and Python. Table (5.3) gives an overview of some of the pros and cons associated to eac wavelet package. As Table (5.2) indicates, PyWavelet package meets most of the requirements to be recognized as a comprehensive tool to be used in this area. Hence, it would be beneficial to investigate how its functions are used. We use this package to implement the practical guide for applying wavelets. Other packages which are not mentioned here falls under application-specific categories and have been designed to address special application requirements.

PyWavelet: PyWavelets is an open source wavelet transform software for Python. It supports multi-dimensional DWT and 1D CWT with single and double precision calculations. data and contain a rich list of mother wavelets. It also support wavelet decomposition and reconstruction.

SciPy.wavelet: SciPy package in Python provides an API which can be used to apply continues wavelets on multi-dimensional data. It contains Morlet and Daubechies wavelets.

Wavelet: Is a wavelet toolkit in R, containing utilities for computing, visualizing for maximal overlap discrete wavelet transformations (MODWT), discrete wavelet transforms (DWT), and their inverses.
Additionally, it has tools for computing and displaying wavelet transform filters, which are employed in the multi - resolution studies and the aforementioned decompositions.

Waveslim: A package in R which implements basic wavelet functionalities to be applied on time series (1D), images (2D), and arrays using fundamental wavelet
3-D analysis.

WaveletComp: Cross-wavelet and phase-difference analysis of time series, alongside with simulation techniques, and wavelet analysis and reconstruction of time series.

	Packages				
	PyWavelet	***SciPy.wavelet***	***Wavelet***	***Waveslim***	***WaveletComp***
Criteria	General Features				
Documentation	✓	✓	✓	✓	✓
Website	✓	✓	✓	✓	✓
Manual	✓	✗	✓	✓	✗
Practical Guide	✓	✗	✗	✓	✓
Language	*Python*	*Python*	*R*	*R*	*R*
Can be used in Notebook	✓	✓	✗	✗	✗
Installation Guide	✓	✓	✓	✓	✓
User-friendly Interface	✓	✓	✓	✗	✗
Datasets	*6*	*0*	*4*	*19*	*4*
Criteria	Technical Aspects				
Support of DWT	✓	✗	✓	✓	✗
Support of CWT	✓	✓	✗	✗	✓
Diverse selection of mother wavelets	✓	✗	✗	✓	✗
Support of Vanishing Points	✓	✓	✓	✓	✓
Support of multi-dimensional DWT	✓	✗	✗	✓	✗
Support of multi-dimensional CWT	✗	✓	✗	✗	✓
Criteria	Post-fit Functions				
Coefficient Extraction	✓	✗	✓	✓	✓
Result of low- and high-pass filters	✓	✓	✓	✓	✓
Plotting	✓	✓	✓	✓	✓

Table 5.3– Comparison of Wavelet Packages. *The table compares the wavelet packages available in Python and R. PyWavelet is increasingly gaining attention as it provides powerful tools with a diverse selection of mother wavelets.*

5.4 Applying Splines in R

In this section a step-by-step tutorial through which one can apply splines on a set of raw functional data is provided. Similar to what was discussed in previous sections, the coding part will be done using Splines package in R. The code for this section can be found in appendix (11).

1. Understanding the Data

   The fist step in functional data analysis is to securitize the data in order to understand what features are present, which columns are needed, specifying the x- and y-axis of for the spline models, check if data is uniformly/irregularly sampled and etc.

2. Loading the Data

   After understanding the data and creating the indexes, it is time to load the data. The dataset can be loaded to the program as a data frame:

```
df <- read_csv('station_32.csv')

x <- df[[1]] #Selecting the X-space
y <- df[[7]] #Specifying the target values
```

3. Specifying Test X-support for the splines

   For the spline models, it is required to specify a test X-space. The index will be used as the domain of the splines. The indexing should be performed with respect to the uniform or irregular nature of the sampling. This can be achieved by using *seq()* function in R. The first two arguments indicate the start and the end of range respectively. The range should be specified based on the number of data points in the dataset. The third argument specify the number of segments we wish to make. The more segments we make, the smoother the curve becomes as it would have more data and the distance between the datapoints becomes smaller.

```
points <- seq(1, 17500, length.out = 17500) #Domain Range of Splines
```

4. Defining the spline models

   Using Splines Package, there are a set of spline models including *bs()*, *ns()* and *smoothing.spline()* available. Depending on what spline is used, there are a set of optional arguments to be passed to the

functions, which we touched upon the most useful ones in the related sections.

Note: the parameters passed to the functions are all intended to produce the results shown in Figure (5.4). One can play around these values to examine how it might effect the smoothness of the curves as we did in previous sections.

```
# defining models

#cubic B-spline
bs.model <- bs(x,df=200)
#B-spline with degree 5
bs5.model <- bs(x,degree=5,df=200)
#Natrual Cubic Spline
ns.model <- ns(x,df=200)
#Smoothing Spline
sp.model <- smooth.spline(y ~ x, cv=FALSE,all.knots=TRUE)
#P-spline
ps.model <- smooth.spline(y ~ x, cv=FALSE,all.knots=FALSE)
```

5. Training

After defining the model, it is time to train the model by passing the training and testing set to the data. The *smoothing.spline()* function which provides smoothing and p-spline methods does not require to use *lm()* function to fit the data and the fitting is done once the model is created. With *cv = TRUE*, the model uses ordinary cross validation (CV) and uses *GCV* if *cv=FALSE*. Figure (5.1) shows the curve produced by the smoothing spline with CV.

```
#Training Data
fit.bs <- lm(y ~ bs.model)
fit.bs5 <- lm(y ~ bs5.model)
fit.ns <- lm(y ~ ns.model) #natural spline
```

6. Predicting

By passing the trained model and test data created in step 3 to *predict()* function, the final curve will be generated.

```
#Predicting the Values
res.bs <- predict(fit.bs, data.frame(x=points))
res.bs5 <- predict(fit.bs5, data.frame(x=points))
res.ns <- predict(fit.ns, data.frame(x=points))
```

7. Plotting the data

As the final step, we can generate figures using *plot()* function so the curves can also be visually analyzed. *lines()* also add the subsequent

plots to the same figure. Figure (5.4) illustrates the result of the experiment.

```
#Plotting the Output
dev.new(width=7, height=4)
plot(x, y, ylab="Available Bikes", xlab="Time", axes=FALSE,ylim=c(0,32))
axis(2); axis(1); box()
lines(points,res.bs , col=2, lwd=2)
lines(points,res.bs5 , col=3, lwd=2) #Green
lines(points,res.ns , col=4, lwd=2) #Dark Blue
lines(sp.model, col=5, lwd=2) #Light Blue
lines(ps.model, col=6, lwd=2) #Perpul
```

5.4.2 Reproducing the Result of Quantile Knot Placement

The visual comparison of the two methods mentioned above can be produced using *mgcv* package in R. We first define the domain of the functional, the function itself and associated noise and the domain of the function.

```
set.seed(0); x <- sort(rnorm(400, 0, pi))
set.seed(1); e <- rnorm(400, 0, 0.4)
y0 <- sin(x) + 0.2 * x + cos(abs(x))
y <- y0 + e
```

In mgcv gam function fits the model to the data. Function *s()* creates the spline. By setting bs argument in *s()* function to *'cr'*, the splines will be produced based on Quantile Knot placement. If *'bs'* is set, the normal Equidistant Knot placement will be applied.

```
library(mgcv)

#fitting cubic B-spline with Quantile knots
qu_fit <- gam(y ~ s(x, bs = 'cr', k = 20))
qu_knots <- qu_fit$smooth[[1]]$xp  #extract knots locations

#fitting cubic B-spline with equidistant knots
eq_fit <- gam(y ~ s(x, bs = 'bs', k = 20))
eq_knots <- eq_fit$smooth[[1]]$knots #extract knots locations
```

The result can be plotted using the snippet below. (see section 3 for the result)

```
#summary plot
dev.new(width=8, height=4)
par(mfrow = c(1,2))
plot(x, y, col= "grey", main = "Quantile Knot Placement");
lines(x, qu_fit$linear.predictors, col = 2, lwd = 2)
abline(v = qu_knots, lty = 2)
plot(x, y, col= "grey", main = "Equidistant Knot Placement");
lines(x, eq_fit$linear.predictors, col = 2, lwd = 2)
abline(v = eq_knots, lty = 2)
```

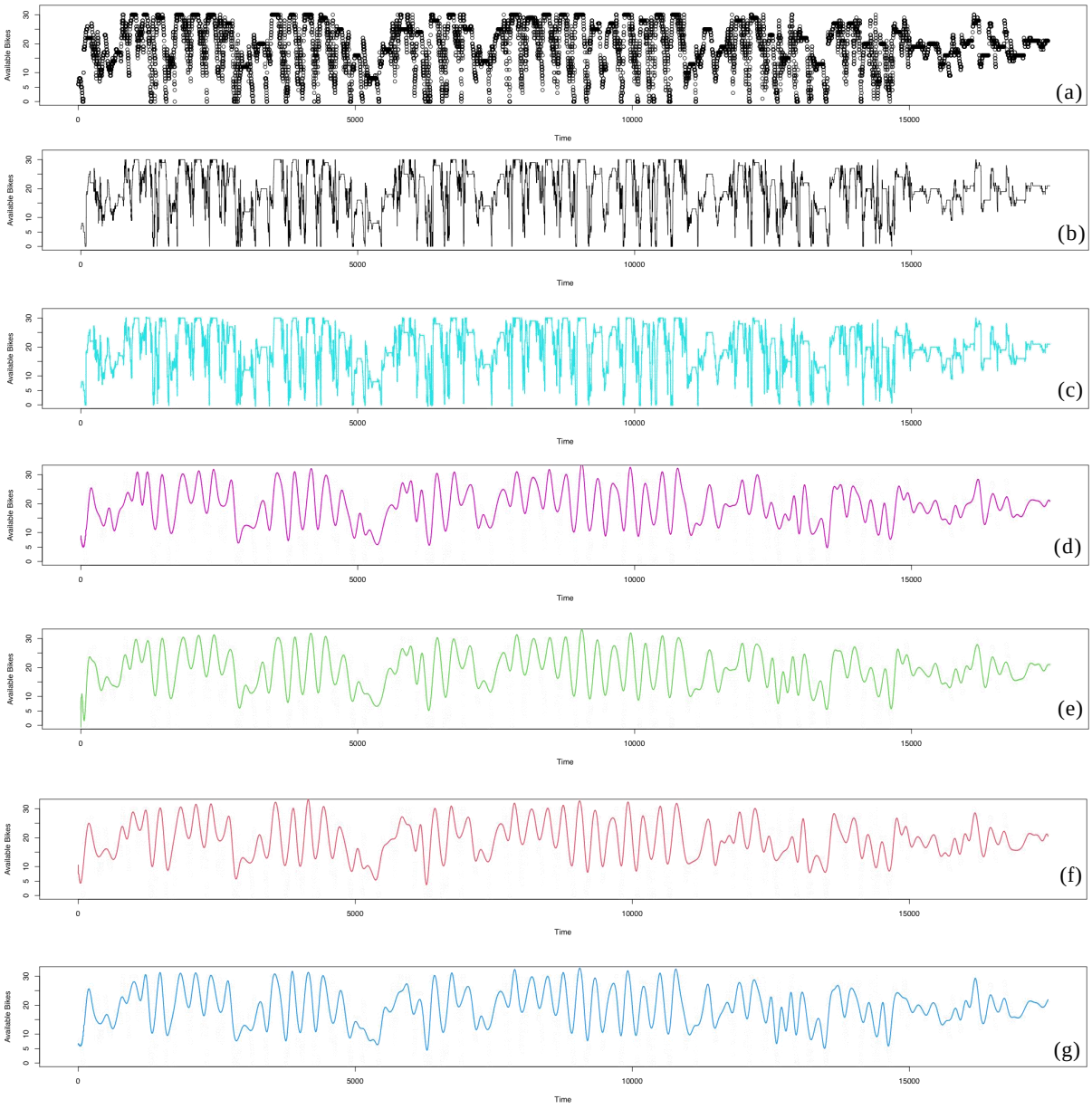


Figure 5.1 – Results of Different Splines on Dublin Bike Data. *(a) and (b) show the Dublin Bike Availability of Pearse Station from Jan 2020 to March 2022 which can be found in [38]. (c), (d), (e), (f) and (g) show the spline fir of the data using Smoothing Spline, P-spline, 5-degree B-spline, Cubic B-spline and Natural Cubic Spline respectively. The degrees of freedom in (e), (f) and (g) is set to 200. The Smoothing and P-spline outperform the other methods.*

Spline	*Degree*	*Degrees of Freedom*	*RMSE*	*Time Taken*
B-spline	*3*	*200*	*0.4553*	*1.19360*
B-spline	*5*	*200*	*0.4498*	*1.25841*
Natural	*3*	*200*	*0.4483*	*1.35437*
Smoothing	*3*	-	*0.0397*	*0.05594*
Panelized	*3*	-	*0.4476*	*0.04451*

Table 5.4 – *Evaluation metrics of using different splines on Dublin Bike Data [38].*

5.5 Applying Wavelet in Python

In this section, we apply DWT denoising techniques on Dublin Bike data to make a smooth fit for the curve.

1. Importing the Package
   We start by importing the Pywavelet package in our python file:

```
import pywt
```

2. Loading the data
   We use pandas *read_csv()* to load the dataset, we then define the X- and Y- space of the data by specifying their corresponding column in the dataset.

```
import pandas as pd

df = pd.read_csv('station32.csv')
x = df[1].values
y = df[7].values
```

3. DWT Function:
   There are four key steps involved in applying DWT for denoising the data:

   A) We need to deconstruct the data by applying dwt. This can be done using *wavedec()* function in pywt package which takes as inputs the data, mother wavelet and the level to which we want to break down the data.

   B) Calculate the threshold for the selected level. Deepening on the level to which we want to decompose the data, the corresponding threshold needs to be calculated. Many have proposed methods through which an optimal threshold can be selected [31, 32, 62, 63], One way is to calculate is based on the absolute deviation of the data. That is what madev() function does.

   C) Keep only coefficients whose value is greater than the threshold.

   D) Then we reconstruct the curve using the remaining coefficients by calling *waverec()* function.

E) The wavelet_denoising() function also take level of decomposition as an input. As discussed in chapter(), larger values for level of decomposition yield more abstract representation of the curve. Since the accuracy of the transformed curve in this experiment is also of interest, we keep the lowest value of the level of decomposition.

```
def madev(d, axis=None):
    #Mean absolute deviation of a signal
    return np.mean(np.absolute(d - np.mean(d, axis)), axis)

def wavelet_denoising(x, wavelet='db4', level=1):
    coeff = pywt.wavedec(x, wavelet, mode="per")
    sigma = (1/0.6) * madev(coeff[-level])
    thresh = sigma * np.sqrt(2 * np.log(len(x)))
    coeff[1:]=(pywt.threshold(i,value=thresh,mode='soft') for i in coeff[1:])
    return pywt.waverec(coeff, wavelet, mode='per')
```

4. Applying DWT Function:
   To see the effect of choosing different mother wavelets, we pass all possible options of DWTs available in pywt package.

```
for wav in pywt.wavelist('discrete_wavelets'):
      filtered = wavelet_denoising(y, wavelet=wav, level=5)
```

5. Plotting the Data:
   Finally we plot the results so it can help us to visually compare the results.

```
plt.figure(figsize=(10, 6))
    plt.plot(y, label='Raw')
    plt.plot(filtered, label='Filtered')
    plt.legend()
    plt.title(f"DWT Denoising with {wav} Wavelet", size=15)
    plt.savefig('Image/'+str(wav)+'.png')
```

Figure(5.2) shows the results of the experiment above. Table (5.6) compares the results based on the root main squared error and time complexity.

The code for this section can be found in appendix (12).

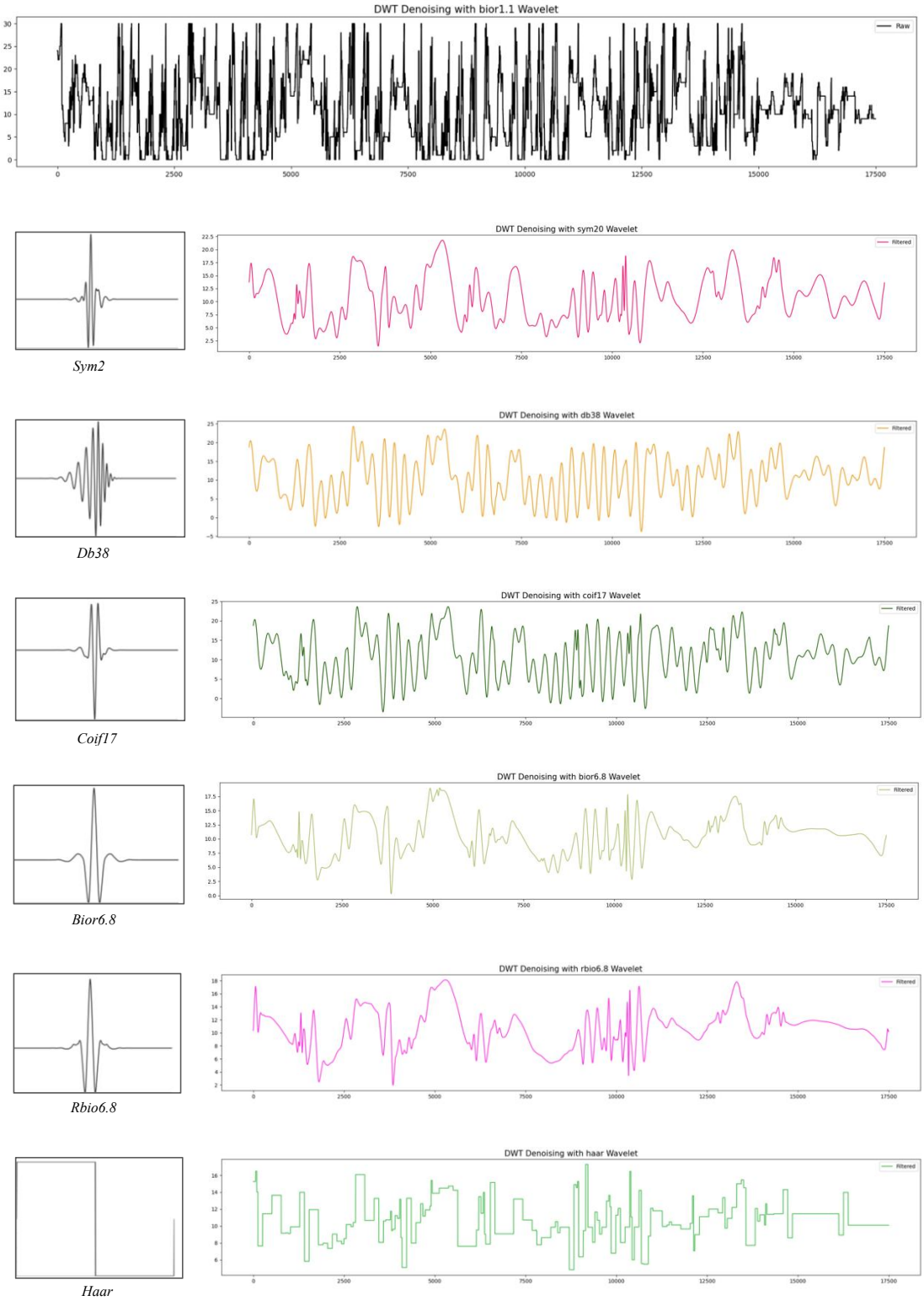


Figure 5.5 – Results of Different Wavelets on Dublin Bike Data. *This is the result of the applying different mother wavelets and vanishing points on Dublin Bike Data. The shape of the mother wavelet used are shown on the left side and can be found in [67]. These are the selection of best fits in terms of accuracy selected after applying cross-validation. The more folding is inside the mother wavelet, the more capable the wavelet becomes to capture local behavior.*

Wavelet	*Vanishing Points*	*RMSE*	*Time Taken*
sym	*20*	*0.5663*	*0.00265*
bior	*6.8*	*0.6070*	*0.00166*
db	*38*	*0.4725*	*0.00647*
coif	*17*	*0.4856*	*0.00578*
rbio	*6.8*	*0.6371*	*0.00166*
haar	*-*	*0.6291*	*0.00130*

Table 5.6 – *Evaluation metrics of using different wavelets on Dublin Bike Data [38].*

5.6 Discussion

In this section, we evaluate the results of the experiment using Splines and Wavelets.

5.6.1 Splines

As it can be observed in Figure(5.1), the smoothing spline which place a knot at every X in the X-space has fully captured the behavior of data. However, it is no longer considered a smooth estimation as it includes the noise of the original data. Table (5.5) compares the Mean Squared Error of the models alongside with their time omplexity. Based on the these values, the P-spline would be a good candidate to be applied on the dataset.

It is notable that the flexibility of the smoothing or P-spline can also be achieved using regression B-spline. However, the burden of handling complexity and flexibility will fall on the shoulder of knot placement. Hence, the computational complexity of regression splines grows exponentially as the degrees of freedom increases. On the other hand, in smoothing splines, the smoothness is controlled by the penalty component and the computational cost is mitigated. To gain more insight, the time consumed by the regression B-spline with 200 knots is equal to time taken by smoothing spline which its equivalent of degrees of freedom is 9450. Although it can be due to the implementation of spline functions in Spline package, however, the findings are in line with what other studies has indicated.

Spline bases work well to simulate smooth curves. For a rough approximation of smooth sample curves correctly observed, cubic spline interpolation with a B-spline basis might be taken into consideration. On the other hand, reconstruction of the correct functional form of noisy smooth curves is adequate for least squares approximation using B-spline basis.

5.6.2 Wavelet

Figure (5.5) illustrates the results of the smoothing experiments using wavelets. The visual format of the mother wavelet used for each curve is also shown in Figure(5.5). Having observed the visual results, it can see that how different selection of mother wavelets affect the final curve. The more folding the mother wavelet contains, the more powerful the transform becomes to capture the local behavior of the data. The compatibility between the shape of the mother wavelet and the data pattern also plays an important role in obtaining a smooth curve. The number of vanishing points also make the mother wavelet appear with more folding, adding to the smoothness of the final curve, Therefore, we used the maximum value of vanishing points for the discrete mother wavelets used in this experiment.

As Table (5.6) suggests, Daubechies with 38 vanishing points yields the lowest mean squared error. The visual result in Figure (5.5) supports this fact. The order of time complexity has been reported the same with haar being the lowest. Having considered the accuracy, time complexity and the form of the final curve, the db38 is a good compromise for the given data set.

5.6.3 Splines and wavelets

The best spline fine for the Dublin Bike data turned out to be P-splines. P-splines is a smoothing spline which has cubic B-spline representation and utilizes less number of knots compared to smoothing splines. db38 mother wavelet was also chosen from the wavelet transforms.

5.6.3.1 Accuracy

The accuracy of both spline fits and wavelet transform are of the same order with smoothing spline being the lowest in splines and *db38* being the lowest in wavelets. However, since the smoothing spline is not as smooth as desired, we compare the P-spline from splines with *db38* from wavelets.

5.6.3.2 Time

Although the Root Mean Squared Errors (RMSE) of the splines and wavelets were close to each other, however, the time taken for fitting the curve in splines is considerably of higher orders. This indicates that spline fitting for densely-sampled data could be computationally complex. One justification for this is that the P-spline place a large number of knots, and

then fit a cubic polynomial line between the each pair of knots. It also impose a penalty for the adjacent cubic lines to control the level of smoothness. This process becomes costly when the sampling is densely made and the number of knots grows. On the other hand, Wavelets deconstruct the data by convolving the mother wavelet to the data with incrementing discrete scales, which is an arithmetic operation. It then calculate the threshold for each level of decomposition and filter out the data which lies beyond the threshold. In our case, we decomposed only to one level so it can hold the main characteristics of the data. Otherwise, only an abstract approximation of the data would have been obtained. (see Figure (4.5(e)) Performing an arithmetic operation on the data is considered significantly less complex compared to spline fitting. Therefore we see that the time taken for wavelet transforms are outstandingly lower than the splines. Therefore, as many suggest, the wavelets would be preferred to deal with noisy data compared to splines.

Chapter 6

Conclusion

In this dissertation, we reviewed Splines and Wavelets as Smoothing Techniques in the context of FDA. We started the paper with a review of the fundamental aspects of Functional Data Analysis (FDA). FDA offers new techniques through which additional information from the data can be extracted. This can be done by approximating a smooth function to describe the behavior of the data. evaluating the underlying patterns exist in the data e.g. derivatives. The function must represent the dominant characteristics of the original data. FDA treats the data as a functional object. To create this function, we require some smoothing methods including Splines, Wavelets. Although they are different tools for a common goal but they are fundamentally different.

We discussed different families of Splines and basis functions including Regression Splines and Smoothing Splines. The computational complexity and the level of smoothness in Regression Splines are proportional to the

number of knots and the degrees of freedom. Splines in nutshell utilizes a set of knots placed on the X-spaces and fit polynomial lines between the knots to make a smooth polynomial curve. This method has the advantage that the computational complexity reduces to the estimation of the coefficients of the estimation function.

We compared different methods of knot placement including equidistant, quantile, and data-driven knot placement on a set of functional data. It turned out that equidistant knot placement performs well in most of the cases. Data-driven knot placement also becomes useful when data is noisy and sampling intervals are not dense.

We also reviewed and evaluated the programming tools to apply smoothing techniques in both Python and R. A practical guide through which the two smoothing techniques can be applied on a set of raw functional data is provided in this dissertation. The code for applying Splines on Dublin Bike Data time series was written in R language. Splines packages in R are the most commonly used tools in the literature. We applied Regression and Smoothing Splines on data. The result of the experiment showed that P-spline was the best compromise between the smoothness, capturing local behavior, accuracy and time complexity.

In addition, we applied wavelet denoising using Discrete Wavelet Transforms using *PyWavelet* Package in Python. The choice of mother wavelet, level of decomposition and the frequency threshold identified as the key factors contributing to the level of smoothness of the output curve. There are ongoing studies to find optimal choice for the parameters mentioned above mainly based on cross-validation. For Dublin Bike data, db38 with one level of decomposition produced the best result. The results of the experiments indicated that Wavelet smoothing techniques outperforms Splines in terms of time complexity even though they provide the same amount of accuracy. However, this could vary if the level of decomposition increases.

As the main goal of this dissertation has been to be read both as a review and a manual, we attempted to cover fundamental aspects of the two smoothing techniques to be used in FDA and provided practical experiments throughout each section as well as a separate chapter dedicated to implementations of Splines and Wavelets using R and Python.

We strongly hope that the findings of this research could be useful for future studies carried out in this area.

6.1 Future Scope

There are other areas to be investigated in order to elevate the work presented by this dissertation. Other types of functional datasets such as images or other multi-dimensional data can be examined to address the viability of Splines and Wavelets as a smoothing tool in Functional Data Analysis.

Furthermore, The parameter tuning in both Splines and Wavelets significantly contributes to the level of smoothness of the estimated curve. Hence, it is worth investigating the new techniques which address the drawbacks of the current versions.

Splines package in R and Pywavelet package in Python used to implement Spline smoothing and wavelet smoothing respectively and cover the practical side of this study. Although they provide all the essential requirements of this area, however, there are other libraries e.g. mgcv in R, SciPy in python which also offer a set of tool both for splines and wavelets which can be examined in future studies.

Since the smoothing techniques is the first step in FDA, it would be insightful to see what implication the choice of smoothing technique would have in other aspects of FDA such as FCPA.

Apart from accuracy and time complexity, other categories of evaluation metrics can be introduced to examine the implication of different choices of smoothing techniques. For instance, one can compare Splines and Wavelets based on their ability in clustering functional objects.

References

[1] Wang, Jane-Ling and Chiou, Jeng-Min and Muller, Hans-Georg, Functional Data Analysis (article) Author, vol. 3, Annual Review of Statistics and Its Application, 2016, pp. 257-295.

[2] Levitin, Nuzzo, Vines, and Ramsay, Introduction to Functional Data Analysis, Canadian Psychology/Psychologie canadienne, 2007.

[3] Mas, André, and Besnik Pumo, "Functional linear regression with derivatives," Nonparametric Statistics, vol. 21, no. 1, pp. 19-40, 2009.

[4] Ferraty, Frédéric, Andre Mas, and Philippe Vieu, "Advances on nonparametric regression for functional variables," Australian and New Zealand Journal of Statistics, vol. 49, no. 3, pp. 267-286, 2007.

[5] Ramsay, James O, "When data are functions," Psychometrika, vol. 47, pp. 379-396, 1982.

[6] Ullah and Finch, Applications of functional data analysis: A systematic review, BMC Medical Research Methodology, 2013.

[7] A.M. Aguilera, M.C. Aguilera-Morillo, "Comparative study of different B-spline approaches for functional data," Mathematical and Computer Modelling, vol. 58, no. 7-8, pp. 1568-1579, 2013.

[8] De Boor, Carl, and Carl De Boor, "A practical guide to splines.," Springer-Verlag, vol. 27, 1978.

[9] Eilers, Paul HC, and Brian D. Marx, "Flexible smoothing with B-splines and penalties," Statistical Science, vol. 11, pp. 89-121, 1996.

[10] J. H. Ahlberg, E. N. Nilson, J. L. Walsh, The Theory of Splines and Their Applications, Ahlberg, Ed., Academic Press, 1967, p. 283.

[11] Brumback, Babette A., and John A. Rice, "Smoothing Spline Models for the Analysis of Nested and Crossed Samples of Curves," Journal of the American Statistical Association, vol. 93, no. 443, pp. 961-976, 1998.

[12] Eubank, Randall L, "Nonparametric Regression and Spline Smoothing," CRC Press, 1999.

[13] Racine, Jeffrey S, "A Primer on Regression Splines," http://cranrprojectorg/web/packages/crs/vignettes/splineprimerpdf, 2014.

[14] Hastie, T., Tibshirani, R., Friedman, J. H., & Friedman, J. H., The elements of statistical learning: data mining, inference, and prediction., vol. 2, New York, 2009.

[15] Eilers, Paul HC, and Brian D. Marx, "Splines, Knots, and Penalties," WIREs Computational Statistics, 2010.

[16] Basna, Rani, Hiba Nassar, and Krzysztof Podgórski, "Data driven orthogonal basis selection for functional data analysis," Journal of Multivariate Analysis, vol. 189, p. 104868, 2022.

[17] Maharani, M., and D. R. S. Saputro, "Generalized Cross Validation (GCV) in Smoothing Spline Nonparametric Regression Models," Journal of Physics: Conference Series, vol. 1808, p. 012053, 2021.

[18] Aydın, Dursun, and Memmedağa Memmedli, "Optimum smoothing parameter selection for penalized least squares in form of linear mixed effect models," Optimization, vol. 61, pp. 459-476, 2012.

[19] Green, Peter J., and Bernard W. Silverman, Nonparametric regression and generalized linear models: a roughness penalty approach, Crc Press, 1993.

[20] Perperoglou, Aris, Willi Sauerbrei, Michal Abrahamowicz, and Matthias Schmid, "A review of spline function procedures in R," BMC Medical Research Methodology, vol. 19, no. 1, p. 46, 2019.

[21] Ramsay JO, Silverman BW, "Functional data analysis," Springer, 2005.

[22] L. Chun-Lin, "A tutorial of the wavelet transform," NTUEE, 2010.

[23] Torrence, Christopher, and Gilbert P. Compo, "A practical guide to wavelet analysis," Bulletin of the American Meteorological society, vol. 79, no. 1, pp. 61-78, 1998.

[24] Polat, Cigdem, and Mehmet Siraç Özerdem, "Introduction to Wavelets and their applications in signal denoising," Bitlis Eren University Journal of Science and Technology, vol. 8, no. 1, pp. 1-10, 2018.

[25] "A guide for using the Wavelet Transform in Machine Learning," ML Fundamentals, 2018. [Online]. Available: https://ataspinar.com/2018/12/21/a-

guide-for-using-the-wavelet-transform-in-machine-learning/. [Accessed 7 August 2022].

[26] Strömbergsson, Daniel, Pär Marklund, Kim Berglund, Juhamatti Saari, and Allan Thomson, "Mother wavelet selection in the discrete wavelet transform for condition monitoring of wind turbine drivetrain bearings," Wiley Online Library, 7 August 2019.

[27] Jang, Young In, Jae Young Sim, Jong-Ryul Yang, and Nam Kyu Kwon, "The Optimal Selection of Mother Wavelet Function and Decomposition Level for Denoising of DCG Signal," Sensors, vol. 21, p. 1851, 2021.

[28] Rafiee, J., P. W. Tse, A. Harifi, and M. H. Sadeghi, "A novel technique for selecting mother wavelet function using an intelli gent fault diagnosis system," Expert Systems with Applications, vol. 36, pp. 4862-4875, 2009.

[29] Pigoli, Davide, and Laura M. Sangalli, "Wavelets in functional data analysis: estimation of multidimensional curves and their derivatives," Computational Statistics & Data Analysis, vol. 56, no. 6, pp. 1482-1498, 2012.

[30] Mannelli, Andrea, Francesco Papi, George Pechlivanoglou, Giovanni Ferrara, and Alessandro Bianchini, "Discrete Wavelet Transform for the Real-Time Smoothing of Wind Turbine Power Using Li-Ion Batteries," Energies, vol. 14, no. 2184, 2021.

[31] Srivastava, Madhur, C. Lindsay Anderson, and Jack H. Freed, "A New Wavelet Denoising Method for Selecting Decomposition Levels and Noise Thresholds," Practical Innovations, Open Solutions, vol. 4, pp. 3862-3877, 2016.

[32] Berkner, Kathrin, and Raymond O. Wells Jr, "Smoothness estimates for soft-threshold denoising via translation-invariant wavelet transforms," Applied and computational harmonic analysis, vol. 12, pp. 1-24, 2002.

[33] StatLib, "Tecator Dataset," 2005. [Online]. Available: http://lib.stat.cmu.edu/datasets/tecator. [Accessed 11 August 2022].

[34] Shang, Han Lin, Rob J. Hyndman, and Maintainer Han Lin Shang, "Package 'fds'," CRAN, 31 10 2018.

[35] B. o. Meteorology, "Climate Data Online," Government of Australian, 2022. [Online]. Available: http://www.bom.gov.au/climate/data/. [Accessed 11 August 2022].

[36] "Climate Prediction Center," National Oceanic and Atmospheric Administration, [Online]. Available: https://www.cpc.ncep.noaa.gov/data/indices/. [Accessed 12 August 2022].

[37] Hastie, Trevor, Robert Tibshirani, Jerome H. Friedman, and Jerome H. Friedman, "Datasets for "The Elements of Statistical Learning"," Stanford University, 2009. [Online].
Available: https://hastie.su.domains/ElemStatLearn/. [Accessed 11 August 2022].

[38] D. C. Council, "Dublinbikes DCC," Government of Ireland, 2019. [Online]. Available: https://data.gov.ie/dataset/dublinbikes-api. [Accessed 11 August 2022].

[39] Ramsay, J.O. and Dalzell, C., "Some Tools for Functional Data Analysis", Journal of the Royal Statistical Society: Series B (Methodological), 1991.

[40] Vines, Bradley W., Regina L. Nuzzo, and Daniel J. Levitin, "Analyzing Temporal Dynamics in Music:: Differential Calculus, Physics, and Functional Data Analysis Techniques," Music Perception, vol. 23, no. 2, pp. 137-152, 2005.

[41] Hyndman RJ, Ullah MS, "Robust forecasting of mortality and fertility rates: a functional data approach," *Comput Stat Data Anal,* no. 51, p. 4942–4956, 2007.

[42] A.M. Aguilera and M.C. Aguilera-Morillo, "Penalized PCA approaches for B-spline expansions of smooth functional data," Applied Mathematics and Computation , vol. 219, pp. 7805-7819, 2013.

[43] J. Dauxois and A. Pousse and Y. Romain, "Asymptotic theory for the principal component analysis of a vector random function: Some applications to statistical inference," Journal of Multivariate Analysis, vol. 12, no. 1, pp. 136-154, 1982.

[44] Frédéric FERRATY and Philippe VIEU, "VARIOUS FUNCTIONAL DATASETS," [Online]. Available: https://www.math.univ-toulouse.fr/~ferraty/SOFTWARES/NPFDA/npfda-datasets.html. [Accessed 11 August 2022].

[45] Patrick S. Hagan and Graeme West, "Interpolation Methods for Curve Construction," *Applied Mathematical Finance,* vol. 12, no. 2, pp. 89-129, 2006.

[46] M. E. a. M. Dehghan, "Application of Taylor series in obtaining the orthogonal operational matrix," Computers & Mathematics with Applications, vol. 61, no. 9, pp. 2596-2604, 2011.

[47] J. Fosso-Tande, "Applications of Taylor series," 2013.

[48] E. W. Weisstein, "Fourier Transform," Wolfram Mathworld, [Online]. Available: https://mathworld.wolfram.com/FourierTransform.html. [Accessed 6 August 2022].

[49] Gallagher, Thomas A., Alexander J. Nemeth, and Lotfi Hacein-Bey, "An Introduction to the Fourier Transform: Relationship to MRI," American Journal of Roentgenology, vol. 190, no. 5, pp. 1396-1405, 2008.

[50] G. Wahba, Spline models for observational data, SIAM, 1990.

[51] The Elements of Statistical Learning: data mining, inference, and prediction, vol. 2, Springer, 2009.

[52] Y. Wang, Smoothing splines: methods and application, CRC Press, 2011.

[53] T. C. Lee, "Smoothing parameter selection for smoothing splines: a simulation study," Computational statistics & Data analysis, vol. 42, pp. 139--148, 2003.

[54] A. Wolfer, 3 March 2019. [Online]. Available: https://cran.r-project.org/web/packages/santaR/vignettes/selecting-optimal-df.html. [Accessed 11 August 2022].

[55] Li, Weishi, Shuhong Xu, Gang Zhao, and Li Ping Goh, "Adaptive knot placement in B-spline curve approximation," Computer-Aided Design, vol. 37, no. 8, pp. 791-797, 2005.

[56] W. Klein, "An Evaluation of Knot Placement Strategies for Spline Regression," 2021.

[57] R. Basna, "Data driven orthogonal basis selection for functional data analysis," Github, 7 9 2021. [Online]. Available: https://ranibasna.github.io/ddk/index.html. [Accessed 10 August 2022].

[58] Ismail, Adham R., and Shihab S. Asfour, "Discrete wavelet transform: a tool in smoothing kinematic data," Journal of Biomechanics, vol. 32, no. 3, pp. 317-321, 1999.

[59] P. Yuhua, "Wavelet transform based filter for smoothing of signals," 1999 International Conference on Computational Electromagnetics and its Applications, 1999.

[60] Ngui, Wai Keng, M. Salman Leong, Lim Meng Hee, and Ahmed M. Abdelrhman, "Wavelet Analysis: Mother Wavelet Selection Methods," Applied

Mechanics and Materials, vol. 393 , pp. 953-958, 2013.

[61] Perperoglou, Aris, Willi Sauerbrei, Michal Abrahamowicz, and Matthias Schmid, "Multivariable Regression Modelling: A review of available spline packages in R.," BMC medical research methodology, 2019.

[62] Cai, Chunsheng and Harrington, Peter de B, "Different discrete wavelet transforms applied to denoising analytical data," Journal of chemical information and computer sciences, vol. 38, pp. 1161-1170, 1998.

[63] Olivo, Jean-Christophe, "Automatic threshold selection using the wavelet transform," CVGIP: Graphical Models and Image Processing, vol. 56, pp. 205-218, 1994.

[64] Podgórski, Rani Basna and Hiba Nassar and Krzysztof, "Data driven orthogonal basis selection for functional data analysis," Journal of Multivariate Analysis, vol. 189, p. 104868, 2022.

[65] Ramsay, J.O. and Dalzell, C, "Some tools for functional data analysis," The Royal Statistical Society, vol. 53, no. 3, pp. 539-561, 1991.

[66] D. Gervini, "Free-knot spline smoothing for functional data," Royal Statistical Society, vol. 68, no. 4, 2006.

[67] F. Wasilewski, "WAVELET BROWSER," PyWavelets, 2008. [Online]. Available: http://wavelets.pybytes.com/. [Accessed 12 August 2022].

Appendix - 1

```
x <- seq(0, 1, length=100) #Defining the range
degree <- 3 # degree of series
nknot <- 5 # number of knots
knots <- (1:nknot) / (nknot+1)
x1 <- outer (x, 1:degree, "^")
x2 <- outer (x, knots,">") * outer (x, knots, "-")^degree
basis <- cbind (x1, x2)
matplot(x, basis, 'l', lwd=2, col= 1:ncol(basis),main='Truncated Power Series Basis
Function')
```

Appendix - 2

```
library(splines)
library(readr)
df <- read_csv('triceps.csv')
x <- df[["age"]]
y <- df[["lntriceps"]]
points <- seq(1, 51, length.out = 1000)
dev.new(width=7, height=4)
plot(x, y, ylab="Triceps skinfold thickness in mm (log)", xlab="Age in years")
#define B-spline and its degrees of freedom
fit.bs <- lm(y ~ bs(x,degree=3)) # bspline
final.curve <- predict(fit.bs, data.frame(x=points))
## add fit lines to the plot
plot(x, y, ylab="Triceps skinfold thickness in mm (log)", xlab="Age in years")
lines(points, predict(fit.bs, data.frame(x=points)), col=2, lwd=2)
box()
```

Appendix - 3

```
library(splines)
library(readr)
library(fds)
df <- fridaydemand
x <- fridaydemand$x
y <- fridaydemand$y[,100]
points <- seq(1, 48, length.out = 1000)
dev.new(width=7, height=4)
plot(x, y, ylab="Electricity Demand (Megawatts)", xlab="Half-hour")
#define B-spline and its degrees of freedom
fit.ns <- lm(y ~ ns(x,df=3)) # bspline
#df=3,5,10,20
#degree=3,7
## add fit lines to the plot
plot(x, y, ylab="Electricity Demand (Megawatts)", xlab="Half-hour")
lines(points, predict(fit.ns, data.frame(x=points)), col=3, lwd=2); box()
```

Appendix - 4

```
library(splines)
library(readr)
library(fds)
df <- nirp
x <- nirp$x
y <- nirp$y[,16]
points <- seq(1100, 2498, length.out = 1000)
dev.new(width=7, height=4)
plot(x, y, ylab="Spectrum", xlab="Wavelength")
#define B-spline and its degrees of freedom
fit.ns <- lm(y ~ ns(x,df=20)) # bspline
#df=3,5,10,20
#degree=3,7
# add fit lines to the plot
plot(x, y, ylab="Spectrum", xlab="Wavelength")
lines(points, predict(fit.ns, data.frame(x=points)), col=3, lwd=2)
box()
```

Appendix - 5

```
library('splines')
require(stats); require(graphics)
library(readr)
df <- read_csv('elnino.csv')
x <- df[[3]]
y <- df[[7]]
dev.new(width=7, height=4)
plot(x,            y,            ylab="Absorbances",            xlab="wavelengths",
axes=FALSE,col='white',ylim=c(5,45))
axis(2); axis(1); box()
f <- y ~ x
fit.sp <- smooth.spline(f, cv=FALSE,all.knots=FALSE)
fit.sp1 <- smooth.spline(f, cv=FALSE,all.knots=FALSE)
print(fit.sp)
#smooth_sp <- predict(fit.sp, data.frame(x=points))
lines(fit.sp, col=4, lwd=3)
lines(fit.sp1, col=2, lwd=2)
```

Appendix - 6

```
library(mgcv)
#sample data
set.seed(0); x <- sort(rnorm(400, 0, pi))  ## note, my x are not uniformly sampled
set.seed(1); e <- rnorm(400, 0, 0.4)
y0 <- sin(x) + 0.2 * x + cos(abs(x))
y <- y0 + e
print(x)
## fitting cubic B-spline with quintile knots
qu_fit <- gam(y ~ s(x, bs = 'cr', k = 20))
qu_knots <- qu_fit$smooth[[1]]$xp  ## extract knots locations
print(qu_knots)
## fitting cubic B-spline with equidistant knots
eq_fit <- gam(y ~ s(x, bs = 'bs', k = 20))
eq_knots <- eq_fit$smooth[[1]]$knots  ## extract knots locations
## summary plot
```

```
dev.new(width=8, height=4)
par(mfrow = c(1,2))
plot(x, y, col= "grey", main = "Quantile Knot Placement");
lines(x, qu_fit$linear.predictors, col = 2, lwd = 2)
abline(v = qu_knots, lty = 2)
plot(x, y, col= "grey", main = "Equidistant Knot Placement");
lines(x, eq_fit$linear.predictors, col = 2, lwd = 2)
abline(v = eq_knots, lty = 2)
```

Appendix - 7

```
library(Splinets)
library(DDK)
library(ggplot2)
library(reshape2)
library(RColorBrewer)
library(readr)
library(fds)
library(splines)
# function that helps in plotting the functional data
df_plot_fda <- function(S_data, time_df, s=1, a= 0.8, n_sample = 10){
  # to do
  # check that the length of the time data agree with the dim of the data
  # write an if statment that in case of empty time_df generate a one based on the
dim S_data
  df_plot <- as.data.frame(t(S_data))
  # selecting samples
  df_plot_n <- df_plot[,1:n_sample]
  # add the time var
  df_plot_n$time <- time_df
  df_plot_melt_n <- melt(df_plot_n, id.vars=c("time"))
  # ploting
  plot_n <- ggplot(df_plot_melt_n, aes(x=time, y = value, color=variable)) +
geom_line(size=s, alpha=a) + theme_minimal() + theme(legend.position = "none",
axis.title = element_blank(), axis.text.y = element_blank())
  res <- list(plot_n, df_plot_melt_n)
  return(res)
}
# get the data from the DKK pacakge
data <- Moisturespectrum$y
#Creating Train set
dt = sort(sample(nrow(data), nrow(data)*0.9))
f_data <- t(data[dt,])
#Generating index for data
t_df <- seq(1, 630)
# test data
f_data_test <- t(data[-dt,])
t_df_test <- seq(1, 630)
# ploting
plot <- df_plot_fda(S_data = f_data, time_df = t_df, a = 1, s = 0.5, n_sample = 10)
p <- plot[[1]]
p <- p + scale_color_brewer(palette = "PuOr")  + theme(legend.position = "None")
initial_knots <- c(1,630)
initial_knots
# Argument L specifies the number of knots to be used.
KS <- add_knots(f = f_data, f_v = f_data, knots = initial_knots, L = 15, M = 1)
KS$Fknots
# ploting the reduction of the AMSE
# For train data
df_knots <- data.frame("x" = seq_len(length(KS[[3]])), "Error_reduction" = KS[[3]])
```

```
# For test data
df_knots_test  <-  data.frame("x"  =  seq_len(length(KS[[4]])),  "Error_reduction"  =
KS[[4]])
print('df_knots_test')
print(df_knots_test)
# ploting amse reduction
{
  plot(x =  df_knots_test$x,y =  df_knots_test$Error_reduction,type='p',pch=16, xlim
=  c(0,10),  ylim  =  c(0,0.01),ylab='Average  Mean  Square  Error',xlab='Iterations',
bty="n", col="deepskyblue4")
  lines(x    =        df_knots$x,y=    df_knots$Error_reduction,type='p',    pch=16,
col='darkorange3')
  legend("topright", legend = c("Training data", "Validation data"),
  col = c("darkorange3", "deepskyblue4"), lty = 3:3, cex = 0.8)
}
# ploting
plot_dist_knots <- df_plot_fda(S_data = f_data, time_df = t_df)
plot_dist_knots[[1]]    +  geom_vline(xintercept  =  KS[[1]],  linetype="dotted")  +
scale_color_brewer(palette = "PuOr")  + theme(legend.position = "None")
#Specifying the range
points <- seq(1, 701)
x <- Moisturespectrum$x
x <- x - 1100
x <- x/2
yi <- Moisturespectrum$y[,2]
dev.new(width=10, height=4)
plot(x, yi, ylab="Absorbances", xlab="wavelengths", axes=FALSE,type='l')
axis(2); axis(1); box()
#Creating the models
fit.DDKbs <- lm(yi ~ bs(x,knots=KS$Fknots))
fit.EQbs <- lm(yi ~ bs(x,df=18))
# Plotting the results
lines(points, predict(fit.EQbs, data.frame(x=points)), axes=FALSE,col='blue', lwd=2)
lines(points, predict(fit.DDKbs, data.frame(x=points)), axes=FALSE,col='red', lwd=2)
```

Appendix - 8

```
library(Splinets)
library(DDK)
library(ggplot2)
library(reshape2)
library(RColorBrewer)
library(readr)
library(splines)
require(stats); require(graphics)
# function that helps in plotting the functional data
create_plot <- function(S_data, time_df, s=1, a= 0.8, n_sample = 10){
  # write an if statment that in case of empty time_df generate a one based on the
dim S_data
  df_plot <- as.data.frame(t(S_data))
  # selecting samples
  df_plot_n <- df_plot[,1:n_sample]
  # add the time var
  df_plot_n$time <- time_df
  df_plot_melt_n <- melt(df_plot_n, id.vars=c("time"))
  # ploting
  dev.new(width=7, height=4)
```

```
  plot_n <- ggplot(df_plot_melt_n, aes(x=time, y = value, color=variable)) +
geom_line(size=s, alpha=a) + theme_minimal() + theme(legend.position = "none",
axis.title = element_blank(), axis.text.y = element_blank())
  res <- list(plot_n, df_plot_melt_n)
  return(res)
}
# get the data from the DKK pacakge
data <- read_csv('Phoneme.csv')
dt = sort(sample(nrow(data), nrow(data)*.9))
f_data <- data[dt,]
t_df <- seq(1, dim(f_data)[2])
# test data
f_data_test <- data[-dt,]
t_df_test <- seq(1, dim(f_data_test)[2])
# ploting
plot <- create_plot(S_data = f_data, time_df = t_df, a = 1, s = 0.5, n_sample = 2)
p <- plot[[1]]
p <- p + scale_color_brewer(palette = "PuOr")  + theme(legend.position = "None")
initial_knots <- c(0, dim(f_data)[2])
KS <- add_knots(f = f_data, f_v = f_data_test, knots = initial_knots, L = 20, M = 1)
KS$Fknots
# ploting the reduction of the AMSE
#For train data
df_knots <- data.frame("x" = seq_len(length(KS[[3]])), "Error_reduction" = KS[[3]])
# For test data
df_knots_test <- data.frame("x" = seq_len(length(KS[[4]])), "Error_reduction" =
KS[[4]])
#Printing the predicted knots
print('df_knots_test')
print(df_knots_test)
# ploting amse reduction
{
  plot(x =  df_knots_test$x,y =  df_knots_test$Error_reduction,type='p',pch=16, xlim
= c(0,10), ylim = c(0,4),ylab='Average Mean Square Error',xlab='Iterations', bty="n",
col="deepskyblue4")
  lines(x    =        df_knots$x,y=    df_knots$Error_reduction,type='p',    pch=16,
col='darkorange3')
  legend("topright", legend = c("Training data", "Validation data"),
  col = c("darkorange3", "deepskyblue4"), lty = 3:3, cex = 0.8)
}
#ploting
plot_dist_knots <- create_plot(S_data = f_data, time_df = t_df)
plot_dist_knots[[1]]    +  geom_vline(xintercept  =  KS[[1]],  linetype="dotted")  +
scale_color_brewer(palette = "PuOr")  + theme(legend.position = "None")
print(c(KS$Fknots))
#Read the dataframe
df <- read_csv('Phoneme2.csv')
x <- df[[1]] #X-axis
y <- df[[2]] #Y-axis
dev.new(width=7, height=4)
plot(x, y, ylab="Absorbances", xlab="wavelengths", axes=FALSE,type='l')
axis(2); axis(1); box()
#Specifying the range
points <- seq(2, 150, length.out = 150)
lines(x, y, ylab="Absorbances", xlab="wavelengths", axes=FALSE, ylim=c(2,6))
#Creating the models
fit.DDKns <- lm(y ~ bs(x,knots=KS$Fknots)) #natural spline
fit.EQns <- lm(y ~ bs(x,df=24)) #natural spline
#Making predictions
DDK_ns_predictions <- predict(fit.DDKns, data.frame(x=points))
EQ_ns_predictions <- predict(fit.EQns, data.frame(x=points))
# Plotting the results
```

```
lines(points, predict(fit.EQns, data.frame(x=points)), axes=FALSE,col='blue', lwd=1)
lines(points, predict(fit.DDKns, data.frame(x=points)), axes=FALSE,col='red', lwd=1)
```

Appendix - 9

```
import pywt
import warnings
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
warnings.filterwarnings("ignore")
# Reading the file
df_train = pd.read_csv('Phoneme2.csv')
df_train.head()
n_times = 150
data = df_train['signal'][:n_times].values
def rmse(predictions, targets):
    return np.sqrt(((predictions - targets) ** 2).mean())
def mse(predictions, targets):
    return ((predictions - targets) ** 2).mean()
#  Calculates the mean absolute deviation of the data
def madev(d, axis=None):
    return np.mean(np.absolute(d - np.mean(d, axis)), axis)
#  Defining Denoising function with threshold
def wavelet_denoising(x, wavelet='db4', level=1):
    #Decomposing the data
    coeff = pywt.wavedec(x, wavelet, mode="per")
    #Calculating the threshold
    sigma = (1/0.6745) * madev(coeff[-level])
    uthresh = sigma * np.sqrt(2 * np.log(len(x)))
    #Calcluate the approximation coefficients
    coeff[1:] = (pywt.threshold(i, value=uthresh, mode='hard') for i in coeff[1:])
    #Using the approximation coefficients to reconstruct the function
    return pywt.waverec(coeff, wavelet, mode='per')
#sample wavelets than can be applied on the data
waves = ['sym4','sym7','sym11','sym16','sym20','sym4','db19','coif6','bior1.5']
plt.show()
for mother_wav in pywt.wavelist(kind='discrete'):
    filtered = wavelet_denoising(data, wavelet=mother_wav, level=1)
    plt.figure(figsize=(24, 4))
    plt.plot(data, label='Raw')
    plt.plot(filtered, label='Filtered')
    print('rmse: ',mother_wav,',',rmse(filtered, data))
    print('mse: ',mother_wav,',',mse(filtered, data))
    plt.legend()
    plt.title(f"DWT Denoising with {mother_wav} Wavelet", size=15)
    plt.savefig('Image/'+str(mother_wav)+'.png')
```

Appendix - 10

```
import pywt
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
# Reading the imput file
filename = 'station_32.csv'
df = pd.read_csv(filename, header=None)
x = df[1].values
data = df[7].values
```

```
fig, ax = plt.subplots(figsize=(6,1))
ax.set_title("Original Data: ")
ax.plot(data)
plt.show()
data = data
mother_wavelet = 'sym20'
fig, axarr = plt.subplots(nrows=10, ncols=2, figsize=(8,8))
# Specifying the level to which we want to decompose the data
# Larger values will result in abtract representation of data
for level in range(10):
    (data, coeff_d) = pywt.dwt(data, mother_wavelet)
    axarr[level, 0].plot(data, 'r')
    axarr[level, 1].plot(coeff_d, 'g')
    axarr[level,   0].set_ylabel("Level   {}".format(level   +   1),   fontsize=14,
rotation=90)
    axarr[level, 0].set_yticklabels([])
    if level == 0:
        axarr[level, 0].set_title("Approximation coefficients", fontsize=14)
        axarr[level, 1].set_title("Detail coefficients", fontsize=14)
    axarr[level, 1].set_yticklabels([])
plt.tight_layout()
plt.show()
```

Appendix - 11

```
library('splines')
require(stats); require(graphics)
library(readr)
MSE <- function(real_value, predicted) {
  return(mean((real_value - predicted)^2))
}
RMSE <- function(real_value, predicted) {
  return (sqrt(mean((real_value - predicted)^2)))
}
df <- read_csv('station_32.csv')
x <- df[[1]]
y <- df[[7]]
#Defining the range of spline
points <- seq(1, 17500, length.out = 17499)
# defining models
bs.model <- bs(x,df=200) # cubic B-spline
bs5.model <- bs(x,degree=5,df=200) # B-spline with degree 5
ns.model <- ns(x,df=200) # Natrual Cubic Spline
sp.model <- smooth.spline(y ~ x, cv=FALSE,all.knots=TRUE) #Smoothing Spline
ps.model <- smooth.spline(y ~ x, cv=FALSE,all.knots=FALSE) #P-spline
#Training Data
fit.bs <- lm(y ~ bs.model)
fit.bs5 <- lm(y ~ bs5.model)
fit.ns <- lm(y ~ ns.model) #natural spline
#Predicting the Values
res.bs <- predict(fit.bs, data.frame(x=points))
res.bs5 <- predict(fit.bs5, data.frame(x=points))
res.ns <- predict(fit.ns, data.frame(x=points))
#Plotting the Output
dev.new(width=24, height=4)
plot(x, y, ylab="Available Bikes", xlab="Time",type='l',axes=FALSE,ylim=c(0,32))
axis(2); axis(1); box()
#Calculating the RMSE
print('Cubic B-Spline:')
print(RMSE(y,res.bs))
```

```
print('5th order B-Spline:')
print(RMSE(y,res.bs5))
print('Natrual Spline:')
print(RMSE(y,res.ns))
#Plotting the results
#Comment each you do not wish to display
lines(points,res.bs , col=2, lwd=2) #Red
lines(points,res.bs5 , col=3, lwd=2) #Green
lines(points,res.ns , col=4, lwd=2) #Dark Blue
lines(sp.model, col=5, lwd=2) #Light Blue
lines(ps.model, col=6, lwd=2) #Perpul
```

Appendix - 12

```
import pywt
import warnings
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import time
warnings.filterwarnings("ignore")
def rmse(predictions, targets):
    return np.sqrt(((predictions - targets) ** 2).mean())
def mse(predictions, targets):
    return ((predictions - targets) ** 2).mean()
# Reading hte fule
filename = 'station_32.csv'
df = pd.read_csv(filename, header=None)
function = df[7].values
#  Calculates the mean absolute deviation of the data
def madev(d, axis=None):
    return np.mean(np.absolute(d - np.mean(d, axis)), axis)
#  Defining Denoising function with threshold
def wavelet_denoising(x, wavelet='db4', level=1):
    #Decomposing the data
    coeff = pywt.wavedec(x, wavelet, mode="per")
    #Calculating the threshold
    sigma = (1/0.6) * madev(coeff[-level])
    uthresh = sigma * np.sqrt(2 * np.log(len(x)))
    #Calcluate the approximation coefficients
    coeff[1:] = (pywt.threshold(i, value=uthresh, mode='soft') for i in coeff[1:])
    #Using the approximation coefficients to reconstruct the function
    return pywt.waverec(coeff, wavelet, mode='per')
# Applying discrete wavelets on the input data
# Cross-validation based on Mean Squred Error
for wav in pywt.wavelist(kind='discrete'):
    start = time.time()
    #Specifying the level to which we intend to decompse the data
    filtered = wavelet_denoising(function, wavelet=wav, level=5)
    end = time.time()
    col = (np.random.random(), np.random.random(), np.random.random())
    plt.figure(figsize=(24, 4))
    plt.plot(function, label='Raw',color='black')
    plt.plot(filtered, label='Filtered',color=col)
    print(wav,',',mse(filtered, function))
    print(wav,',',end-start)
    plt.legend()
    plt.title(f"DWT Denoising with {wav} Wavelet", size=15)
    plt.savefig('Image/'+str(wav)+'.png')
```