**Chapter 1: Introduction**

Functional data involves estimating functional parameters describing data that are not themselves functional

A theme is using information of derivatives

Consider the problem of estimating a pdf p to describe the distribution of a sample of observations x1,…,xn. If we assumed the pdf was in conventional form we would use some parametric model defined by a mixed and usually small number of parameters. Say you didn’t want to assume a distribution. Non parametric density estimation methods assume only smoothness and permit much flexibility in the estimated as the data requires. The number of parameters is not fixed in advance and attention is put on estimating and not on estimating parameters

Linear operator is:

Assuming that a functional datum for replication arrives as a finite set of measured values then first task is to convert these values to a function with values computable for any desired t. If these observationsa re assumed to be errorless then this process is *interpolation* but if there is any observational error that needs removing the conversion from dat to function is smoothing.

Fourier series used for data with periodicity eg angles

**Chapter 3: How to Specify Basis Systems For Building Functions**

Basis notation:

For replication (eg station, person) with K basis functions then function for data is written as :

Also Written as:

Create…….basis functions create basis for functional approximation. Can make constant basis which uses basis function 1 and monomial which uses as basis functions.3

Create.bspline.basis and create.fourier.basis do as expected.

B-splines

Order of polynomial is one higher than its degree. E.g. straight line is of order 2 as it contains a constant term

The order of each polynomial segment is the same for each subinterval.

Each break point can have several knots. The number of knots determines how many derivatives must match for the neighbouring polynomials on each side of the break point. If one knot is placed then the number of matching derivatives must be two less than its order.

Typically one knot per break point expect the end points. They get as many knots as the order of the spline so that function drops to 0 outside of range

Number of basis functions = order + number of interior knots

Create.bspline.basis(c(0,2\*pi),7,4) creates a bspline basis across 0 to which has 7 break points and order 4 basis functions.

Useful to remember to fix the order of the spline basis to be at least two higher than the highest order derivative used. So say you have 31 data points that are ages ranging from 1 to 18 want to position a knot at each of these sampling points and want to use order 6 splines. To do this use relation above to give 29+6=35. 29 as looking at interior knots so exclude last two data points. Then function used would be create\_bspline\_basis(c(1,18),35,6,breaks=age).

Need to redefine x axis so length of each subinterval is equal to one or more.

**Chapter 4: How to Build Functional Data Objects**

Once we have a basis only need to provide a coefficient vector of length K for each function we wish to define.

If only one function is defined the coefficients are loaded into a vector of length K (or a matrix with K rows and 1 column)

If N functions are needed say for a sample of functional observations of size N, we arrange the coefficient vectors in a K by N matrix.

If functions are 3 dimensional for example for 3 cooridinates then load in K by N by m 3D array.

So order is number of basis functions, number of functions or functional observations, number of dimensions of the functions.

Adding labels to functional data is a good way of providing info for graphical representations.

Need:

Label for x axis eg Time, Day

Label for replication dimension eg weather station, person or child

Label range eg temperature, angle etc

Provide in a label vector of length 3

We may also want each replication dimensions/the range to have its own labels, eg the name of each weather station. Labels for replications and variables can be supplied as lists within lists such that weather station is name of a list in the labels vector and that list contains the names of each weather station

Methods for Functional Data objects:

Can add : fdobj1 + fdobj2 or fdobj + a

Can subtract: fdobj1 - fdobj2 or fdobj - a

Can Multiply: fdobj1 \* fdobj2 or fdobj \* a

Can raise to a power: fdobj^a (Can produce errors if a is negative)

a is a scalar.

The multiplication and exponentiation of these objects is not as expected.

Mean of a set of functions is achieved by: mean(fdobj)

Often want to work with a set of values of a function at a specified value t stored in a vector tvec.

Eval.fd does this and can also evaluate the data at derivatives using

Use standard plot functions to plot functional data objects. Can control line colour etc through that. Can use lines to add lines to fd data

For smoothing using regression analysis:

Compute coefficients for functional data object using the usual equation for regression coefficients

Basismat=eval.basis(x\_axis\_obs,basis)

To get coeffiecients in R then  **=** crossprod(basismat) and  **=** crossprod(basismat,observed\_data) and so coefficients crossprod()

Plotfit.fd plots the fit of a functional data object if original dat is supplied

**Linear Differential Operators**

refers to a linear differential operatorrefers to the application of a linear differential operator to a function x. In general

Where are either constants or functions

Lfd gives a linear differential operator in R:

Nderiv gives the highest order derivative in the operator

Bwtlist is a list that contains the coefficient functions defining the operator. If a coefficient function varies over t then these will be functional data objects with a single replication.

Int2Lfd(n) gives operator

Vec2Lfd(c(1,1,0),c(0,365)) gives operator

**Bivariate Functional Data Objects**

The availability of a sample of N curves make sus wonder how they vary among themselves. The analogue of the correlation and covariance matrices in multivariate context are the correlation and covariance functions and . Also need to define functions of two arguments.

Certai types of bivariate regression require bivariate regression coefficient functions. The bivariate functional data class with name bifd does this.

It requires two basis systems and a matrix of coefficients for a single object. In mathematical notation an estimate is:

**Chapter 5: Smoothing**

**Regression Splines: Smoothing By Regression Analysis**

Tend to define data fitting as minimising the sum of squares

In this case SSE becomes

Motivated by error model:

Can be written as in matrix form:

And least squares estimate of coefficient vector c is:

Use smooth.basis to get coeffiecients:

Needs 3 arguments: 1st the values found in data of the x axis, then corresponding values on y axis then the basis being used.

This function outputs a list containing 3 things: a functional data object with coefficients of interest ($fd), the degrees of freedom of the fitted curves ($df), and the value of the generalised cross validation criterion: a measure of lack of fit discounted for degrees of freedom. If there are multiple curves then a vector is returned containing multiple gcv values for each curve

This only works if the number of smoothing functions is significantly less than the number of sampling points.

**Data Smoothing With Roughness Penalties**

For a large number of basis functions possibly more than the number of observations we can use roughness penalising data smoothing.

A popular method is to use called the curvature of x at t. A measure of a functions roughness is the *integrated squared second derivative*  or *total curvature* given by:

Depending on what derivative we are interested in can use different penalties. is used for penalising the roughness of the initial function where as

Is used to penalise its second derivative so it appears smooth.

Whatever roughness term we use we add some multiple of it to the error sum of squares to define the compound fitting criteria. EG using gives us:

The smoothing parameter specifies the emphasis on the second term penalizing curvature relative to goodness of fit quantified in the sum of squared residuals in the first term. With increasing moves closer to a straight line fit and with leaves function to fit as freely to the data as possible so can have wild variation.

**The Roughness Penalty Matrix R**

Generral version with L being just a linear operator is:

With we get:

Now define order K symmetric roughness penalty matrix as:

So:

So:

To do in R use eval.penalty which takes two inputs:

Basisobj= A functional basis object of class basisfd

LFobj= A linear differential operator of class Lfd (this is L in above generalisation).

Let be the matrix/vector of fitted values and be the observed values. Again:

Also:

Where df is the effective degrees of freedom of the fit defined by .

Use fdPar to get functional data parameter object. It takes 3 things a basis object, a derivative order or a duifferential operator L and a smoothing parameter . This is then passed to smooth.basis to get the actual object of interest.

**Choosing**

The generalised cross-validation measure (GCV). Criteria is:

Want it to be minimised.

**Positive, Monotone, Density and Other Constrained Functions**

Sometimes want fitted curve to never be negative eg with precipitation or height a 0 measuerment does not make logical sense.

In this case we let

smooth.pos does this transformation for you

**Monotone Smoothing**

Model then becomes:

Where:

Here is the fixed origin for the range of values of t for which the data are being fit. For monotonically increasing functions can be absorbed into . However for monotocnially decreasing functions we keep separate.

Smooth.monotone does this transformation for you.

**Chapter 6: Descriptions of Functional Data**

**Some functional Descriptive Statistics**

The sample mean and variance functions are as follows:

There is also the bivariate covariance function:

**Functional Probes**

A probe is a tool for highlighting specific variation. Probes are variably weighted linear combinations of function values. Let be a weight function that we apply to a function as follows:

If has been structured so as to be a template for a specific feature of the variation of x the resulting probe value . Does not need to integrate to 0

Inprod does this for us. Use two functional data objects within. First one is and second is .

**Phase Plane Plots**

Want to use velocity vs acceleration

Can think of a lot of processes as like a spring where energy goes between potential and kinetic energy. Eg potential energy corresponds to resources and kinetic energy ios production process in full swing.