Chapter 6
Descriptions of Functional Data

This chapter and the next are the exploratory data analysis end of functional data analysis. Here we recast the concepts of mean, standard deviation, covariance and correlation into functional terms and provide R and Matlab functions for computing and viewing them.

Exploratory tools are often the most fruitful when applied to *residual variation* around some model, where we often see surprising effects once we have removed relatively predictable structures from the data. Summary descriptions of residual variation are also essential for estimating confidence regions.

Contrasts are often used in analysis of variance to explore prespecified patterns of variation. We introduce the more general concept of a *functional probe* as a means of looking for specific patterns or shapes of variation in functional data and of providing methods for estimating confidence limits for estimated probe values.

The phase-plane plot has turned out to be a powerful tool for exploring data for harmonic variation, even in data on processes such as human growth where we do not ordinarily think of cyclic variation as of much interest. It is essentially a graphical analogue of a second order linear differential equation. In fact, the phase-plane plot, developed in detail in this chapter, is a precursor to the dynamic equations that we will explore in Chapter 11.

6.1 Some Functional Descriptive Statistics

Let $x_i, i = 1, \ldots, N$, be a sample of curves or functions fit to data. The univariate summaries, the sample mean and variance functions, are as follows:

$$\bar{x}(t) = N^{-1} \sum_i x_i(t) \quad \text{and} \quad s(t) = (N-1)^{-1} \sum_i [x_i(t) - \bar{x}(t)]^2.$$

These are computed, for the log-precipitation data considered in Section 5.3, as follows:

J.O. Ramsay et al., *Functional Data Analysis with R and MATLAB*, Use R,
DOI: 10.1007/978-0-387-98185-7_6,

```
meanlogprec   = mean(logprec.fd)
stddevlogprec = std.fd(logprec.fd)
```

As always in statistics, choices of descriptive measures like the mean and variance should never be automatic or uncritical. The distribution of precipitation is strongly skewed, and by logging these data, we effectively work with the geometric mean of precipitation as a more appropriate measure of location in the presence of substantial skewness.

Beyond this specific application, the functional standard deviation focuses on the intrinsic variability between observations, e.g., Canadian weather stations, after removing variations that are believed to represent measurement and replication error not attributable to the variability between observations. A proper interpretation of the analyses of this section require an understanding of exactly what we mean by `std.fd` and what is discarded in smoothing.

6.1.1 The Bivariate Covariance Function $v(s,t)$

As we indicated in Chapter 1, the correlation coefficient as a measure of association between two functional observations $x_i(s)$ and $x_i(t)$ on the same quantity or *metric* is often less useful than the simpler covariance coefficient, because they share the same measurement scales. Where we want to quantify the association between two functions x and y having different measurement scales, the correlation will still be useful.

The bivariate covariance function $\sigma(s,t)$ specifies the *covariance* between curve values $x_i(s)$ and $x_i(t)$ at times s and t, respectively. It is estimated by

$$v(s,t) = (N-1)^{-1} \sum_i [x_i(s) - \bar{x}(s)][x_i(t) - \bar{x}(t)]. \tag{6.1}$$

For the log-precipitation data the R command is

```
logprecvar.bifd = var.fd(logprec.fd)
```

The result of this command is a bivariate functional data object having two arguments. If we want to look at the *variance-covariance surface*, these commands in Matlab will do the job:

```
logprecvar_bifd = var_fd(logprec_fd);
weektime        = linspace(0,365,53);
logprecvar_mat  = eval_bifd(weektime, weektime,
                            logprecvar_bifd);
surf(weektime, weektime, logprecvar_mat);
contour(weektime, weektime, logprecvar_mat);
```

The following will do essentially the same thing in R:

```
weektime        = seq(0,365,length=53)
```

```
logprecvar_mat  = eval.bifd(weektime, weektime,
                            logprecvar.bifd)
persp(weektime, weektime, logprecvar_mat,
      theta=-45, phi=25, r=3, expand = 0.5,
      ticktype='detailed',
      xlab="Day (July 1 to June 30)",
      ylab="Day (July 1 to June 30)",
      zlab="variance(log10 precip)")
contour(weektime, weektime, logprecvar_mat)
```

In our experience, contour and three-dimensional surface or perspective plots complement each other in the information that they convey, and both are worth doing. Surface plots draw our eye to global shape features, but we need contour plots to locate these features on the argument plane.

Function `var.fd` may also be used to compute the `cross-covariance` between two sets of curves by being called with two arguments, such as in

```
tempprecbifd = var.fd(tempfd, logprec.fd)
```

If the *cross-correlation surface* is needed, however, we would use the function `cor.fd` or its Matlab counterpart `cor_fd`.

The variance of the log precipitation functions is seen in Figure 6.1 as the height of the diagonal running from (0,0) to (365,365). There is much more variation in precipitation in the winter months, positioned in this plot in the middle of the surface, because the frigid atmosphere near polar stations like Resolute has almost no capacity to carry moisture, while marine stations like Prince Rupert are good for a soaking all year round. One is struck by the topographical simplicity of this particular surface, and we will understand this better in the next section.

The R commands

```
day5time = seq(0,365,5)
logprec.varmat = eval.bifd(day5time, day5time,
                    logprecvar.bifd)
contour(day5time, day5time, logprec.varmat,
        xlab="Day (July 1 to June 30)",
        ylab="Day (July 1 to June 30)", lwd=2,
        labcex=1)
```

return the contour plot of the variance surface shown in Figure 6.2. We see that variance across weather stations is about five times as large in the winter than it is in the summer. The action is in winter in Canada!

The documentation for the `surf` and `contour` functions in Matlab describe enhancements over the images visible in Figures 6.1 and 6.2. With R, other perspective and contour functions are available in the `lattice` (Sarkar, 2008) and `rgl` (Adler and Murcoch, 2009) packages. In particular, the `lattice` package is useful for high-dimensional graphics, showing, e.g., how the relationships displayed in Figures 6.1 and 6.2 vary with region of the country. The `rgl` package provides interactive control over perspective plots.

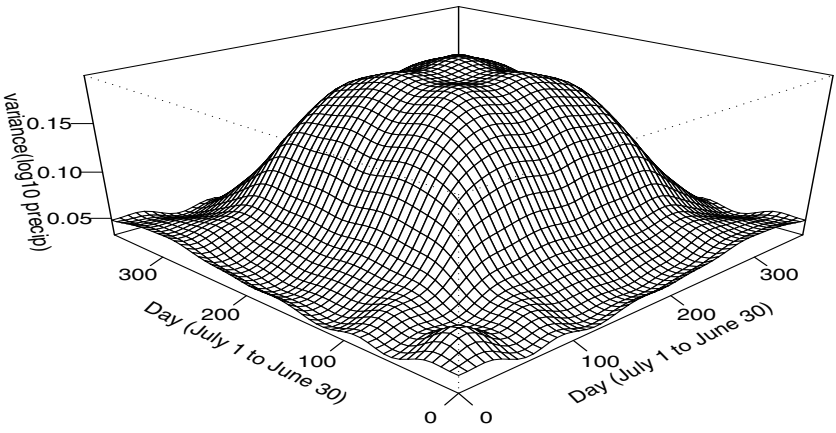

Fig. 6.1 The estimated variance-covariance surface $v(s,t)$ for the log precipitation data.

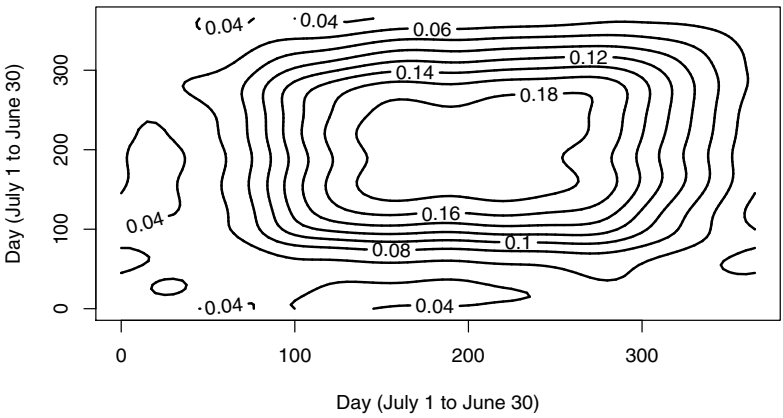

Fig. 6.2 A contour plot of the bivariate variance-covariance surface for the log precipitation data.

6.2 The Residual Variance-Covariance Matrix Σ_e

We considered the question of how the residuals $r_{ij} = y_{ij} - x_i(t_j)$ behave in Section 5.5, and we will return to this question in Chapters 7 and 8. But in the meantime we will need the *conditional covariance matrix* or *residual covariance matrix* describing the covariance of the residuals r_{ij} at argvals t_j, $j = 1, ..., n$. This is an order n symmetric matrix Σ_e. Here the term *conditional* means the variation in the y_{ij}'s left unexplained by a smooth curve or by the use of some other model for the data. We will use this matrix for computing confidence limits for curves and other values.

Unless a large number of replications of curves are available, as is the case for the growth data, we have to restrict our aims to estimating fairly gross structure in the residuals. In particular, it is often assumed that neighboring residuals are uncorrelated, and one only attempts to estimate the standard deviation or variance of the residuals across curves. Figure 5.8 offers a picture of this variation for the log precipitation data. Under this assumption, the order n symmetric matrix Σ_e will be diagonal and will contain values in the vector `logprecvar1` computed in Section 5.5.

We will consider ways of extracting more information Σ_e in Chapter 7.

6.3 Functional Probes ρ_ξ

Purely descriptive methods such as displaying mean and variance functions allow us to survey functional variation without having to bring any preconceptions about exactly what kind of variation might be important. This is fine as far as it goes, but functions and their derivatives are potentially complex structures with a huge scope for surprises, and we may need to "zoom in" on certain curve features.

Moreover, our experience suggests that a researcher seldom approaches functional data without some fairly developed sense of what will be seen. We would be surprised if we did not see the pubertal growth spurt in growth curves or sinusoidal variation in temperature profiles. When we have such a structure in mind, we typically need to do two things: check the data to be sure that what we expect is really there, and then do something clever to look around and beyond what we expect in order to view the unexpected. Chapter 7 is mainly about looking for the dominant modes of variation and covariation, but the tools that we develop there can also be used to highlight interesting but more subtle features.

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

$$\rho_\xi(x) = \int \xi(t)x(t)\,\mathrm{d}t. \tag{6.2}$$

If ξ has been structured so as to be a template for a specific feature or pattern of variation in x, then the resulting probe value $\rho_\xi(x)$ will be substantially far from zero. The term *contrast* in experimental design or linear models has much the same meaning as probe, but there is no particular need for probe functions to integrate to zero.

The value of a probe function is computing using the `inprod` function. Suppose `xifd` and `xfd` are two functional data objects for the weight function ξ and observed curve x, respectively. The probe value `probeval` is computed by the command

```
probeval = inprod(xifd, xfd)
```

The integration in this calculation can be done to within machine precision in many cases, or otherwise is computed by a numerical approximation method.

Probe weight functions ξ may also be estimated from the data rather than chosen a priori. Two methods discussed in Chapter 7, *principal components analysis* and *canonical correlation analysis*, are designed to estimate probes empirically that highlight large sources of variation or covariation.

6.4 Phase-Plane Plots of Periodic Effects

The two concepts of energy and of functional data having variation on more than one timescale lead to the graphical technique of plotting one derivative against another, something that we will call *phase-plane plotting*. We saw an example in Figure 1.15 for displaying the dynamics in human growth.

We now return to the US nondurable goods manufacturing index, plotted in Figures 1.3 and 1.4, to illustrate these ideas. A closer look at a comparatively stable period, 1964 to 1967 shown in Figure 6.3, suggests that the index varies fairly smoothly and regularly within each year. The solid line is a smooth of these data using the roughness penalty method described in Chapter 5. We now see that the variation within this year is more complex than Figure 1.4 can possibly reveal. This curve oscillates three times during the year, with the size of the oscillation being smallest in spring, larger in the summer, and largest in the autumn. In fact, each year shows smooth variation with a similar amount of detail, and we now consider how we can explore these within-year patterns.

6.4.1 Phase-Plane Plots Show Energy Transfer

Now that we have derivatives at our disposal, we can learn new things by studying how derivatives relate to each other. Our tool is the plot of acceleration against velocity. To see how this might be useful, consider the phase-plane plot of the function $\sin(2\pi t)$, shown in Figure 6.4. This simple function describes a basic *harmonic pro-*

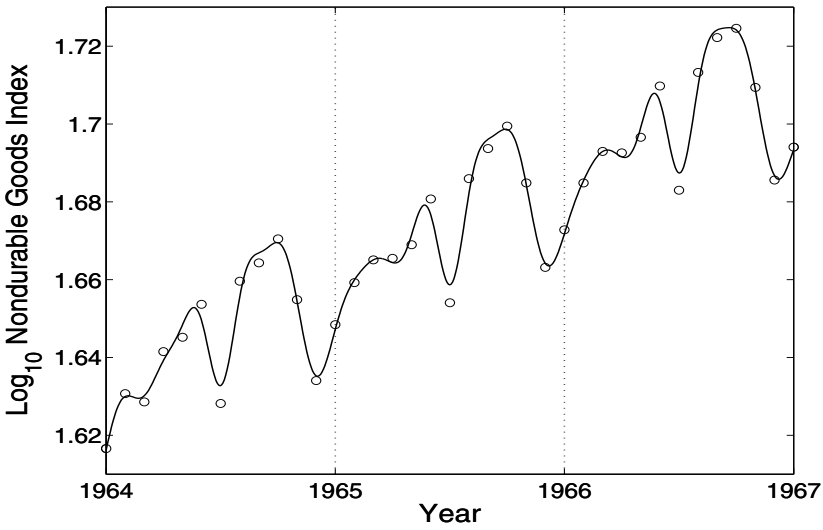

Fig. 6.3 The log nondurable goods index for 1964 to 1967, a period of comparative stability. The solid line is a fit to the data using a polynomial smoothing spline. The circles indicate the value of the log index at the first of the month.

cess, such as the vertical position of the end of a suspended spring bouncing with a period of one time unit.

Springs and pendulums oscillate because energy is exchanged between two states: *potential* and *kinetic*. At times $\pi, 3\pi, \ldots$ the spring is at one or the other end of its trajectory, and the restorative force due to its stretching has brought it to a standstill. At that point, its potential energy is maximized, and so is the force, which is acting either upward (positively) or downward. Since force is proportional to acceleration, the second derivative of the spring position, $-(2\pi)^2 \sin(2\pi t)$, is also at its highest absolute value, in this case about ± 40. On the other hand, when the spring is passing through the position 0, its velocity, $2\pi \cos(2\pi t)$, is at its greatest, about ± 8, but its acceleration is zero. Since kinetic energy is proportional to the square of velocity, this is the point of highest kinetic energy. The phase-plane plot shows this energy exchange nicely, with potential energy being maximized at the extremes of Y and kinetic energy at the extremes of X.

The amount of energy in the system is related to the width and height of the ellipse in Figure 6.4; the larger it is, the more energy the system exhibits, whether in potential or kinetic form.

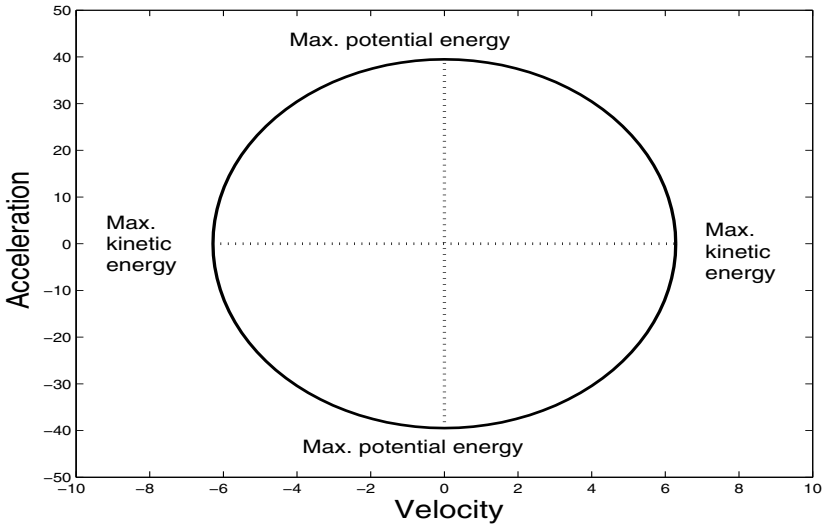

Fig. 6.4 A phase-plane plot of the simple harmonic function $\sin(2\pi t)$. Kinetic energy is maximized when acceleration is 0, and potential energy is maximized when velocity is 0.

6.4.2 The Nondurable Goods Cycles

Harmonic processes and energy exchange are found in many situations besides mechanics. In economics, potential energy corresponds to resources including capital, human resources, and raw material that are available to bring about some economic activity. This energy exchange can be evaluated for nondurable goods manufacturing as displayed in Figure 6.3. Kinetic energy corresponds to the manufacturing process in full swing, when these resources are moving along the assembly line and the goods are being shipped out the factory door.

We use the phase-plane plot, therefore, to study the energy transfer within the economic system. We can examine the cycle within individual years, and also see more clearly how the structure of the transfer has changed throughout the 20th century. Figure 6.5 presents a phase-plane plot for 1964, a year in a relatively stable period for the index. To read the plot, find "jan" in the middle right of the plot and move around the diagram clockwise, noting the letters indicating the months as you go. You will see that there are two large cycles surrounding zero, plus some small cycles that are much closer to the origin.

The largest cycle begins in mid-May (M), with positive velocity and near zero acceleration. Production is increasing linearly or steadily at this point. The cycle moves clockwise through June ("Jun") and passes the horizontal zero acceleration line at the end of the month, when production is now decreasing linearly. By mid-July ("Jly") kinetic energy or velocity is near zero because vacation season is in full swing. But potential energy or acceleration is high, and production returns to the

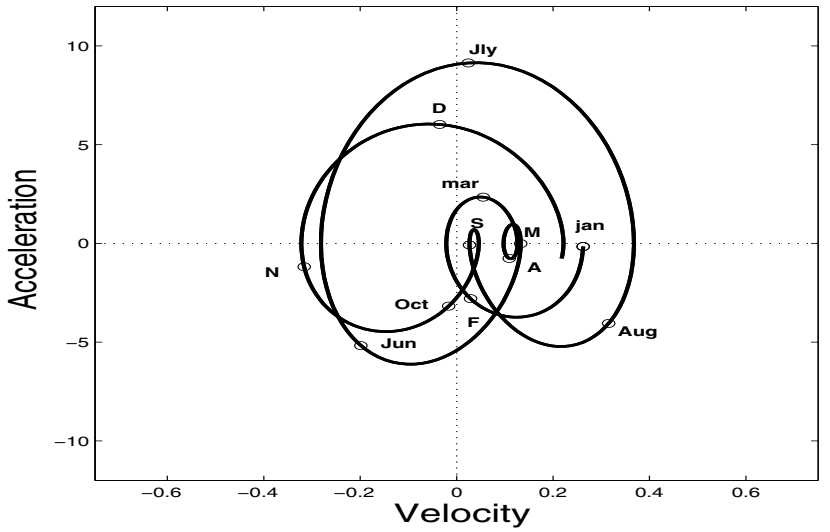

Fig. 6.5 A phase-plane plot of the first derivative or velocity and the second derivative or acceleration of the smoothed log nondurable goods index for 1964. Midmonths are indicated by the first letters or short abbreviations.

positive kinetic/zero potential phase in early August ("Aug"), and finally concludes with a cusp at summer's end (S). At this point the process looks like it has run out of both potential and kinetic energy.

The cusp, near where both derivatives are zero, corresponds to the start of school in September and the beginning of the next big production cycle passing through the autumn months of October through November. Again this large cycle terminates in a small cycle with little potential and kinetic energy. This takes up the months of February and March (F and mar). The tiny subcycle during April and May seems to be due to the spring holidays, since the summer and fall cycles, as well as the cusp, do not change much over the next two years, but the spring cycle cusp moves around, reflecting the variability in the timings of Easter and Passover.

To summarize, the production year in the 1960s has two large cycles swinging widely around zero, each terminating in a small cusplike cycle. This suggests that each large cycle is like a balloon that runs out of air, the first at the beginning of school and the second at the end of winter. At the end of each cycle, it may be that new resources must be marshaled before the next production cycle can begin.

6.4.3 Phase-Plane Plotting the Growth of Girls

Here are the commands in Matlab used to produce Figure 1.15. They use a functional data object `hgtfmonfd` that contains the 54 curves for the Berkeley girls

estimated by monotone smoothing. Velocities and accelerations are first evaluated over a fine mesh of ages for the first ten girls using the `eval_fd` function. Then all 10 phase-plane plots are produced, followed by plots of the sixth girl's curve as a heavy dashed line, and of circles positioned at the age 11.7 for each girl. Labels and axis limits are added at the end.

```
agefine  = linspace(1,18,101);
velffine = eval_fd(agefine, hgtfmonfd(1:10), 1);
accffine = eval_fd(agefine, hgtfmonfd(1:10), 2);
phdl = plot(velffine, accffine, 'k-', ...
            [1,18], [0,0], 'k:');
set(phdl, 'LineWidth', 1)
hold on
phdl = plot(velffine(:,6), accffine(:,6), ...
            'k--', [0,12], [0,0], 'k:');
set(phdl, 'LineWidth', 2)
phdl=plot(velffine(64,index), accffine(64,index), ...
          'ko');
set(phdl, 'LineWidth', 2)
hold off
xlabel('\fontsize{13} Velocity (cm/yr)')
ylabel('\fontsize{13} Acceleration (cm/yr^2)')
axis([0,12,-5,2])
```

What we see is that girls with early pubertal growth spurts, having marker circles near the end of their trajectories, have intense spurts, indicated by the size of their loops. Late-spurt girls have tiny loops. The net effect is that the adult height of girls is not much affected by the timing of the growth spurt, since girls with late spurts have the advantage of a couple of extra years of growth, but the disadvantage of a weak spurt. A hint of the complexity of growth dynamics in infancy is given by the two girls whose curves come from the right rather than from the bottom of the plot.

6.5 Confidence Intervals for Curves and Their Derivatives

indexderivatives!confidence intervals We now want to see how to compute confidence limits on some useful quantities that depend on an estimated function x that has, in turn, been computed by smoothing with a roughness penalty a data vector $\mathbf{y}$. For example, how precisely is the function value at t, $x(t)$, determined by our sample of data $\mathbf{y}$? Or, what sampling standard deviation can we expect if we re-sample the data over and over again, estimating $x(t)$ anew with each sample? Can we construct a pair of *confidence limits* such that the probability that the true value of $x(t)$ lies within these limits is a specified value, such as 0.95? Displaying functions or their derivatives with pointwise confidence limits is a useful way of conveying how

much information there is in the data used to estimate these functions. See Figure 6.6 below for an example.

More generally, confidence regions are often required for the values of linear probes ρ_ξ defined in (6.2), of which $x(t)$ and $D^m x(t)$ are specific examples.

6.5.1 Two Linear Mappings Defining a Probe Value

In order to study the sampling behavior of ρ_ξ, we need to compute two linear mappings plus their composite. They are given names and described as follows:

1. Mapping `y2cMap`, which converts the raw data vector **y** to the coefficient vector **c** of the basis function expansion of x. If **y** and **c** have lengths n and K, respectively, this mapping is a K by n matrix `y2cMap` such that

$$\mathbf{c} = \texttt{y2cMap}\,\mathbf{y}$$

   where the K by n matrix `y2cMap` was defined in Chapter 5 by either (5.5) or (5.17).
2. Mapping `c2rMap`, which converts the coefficient vector **c** to the scalar quantity $\rho_\xi(x)$. This mapping is a 1 by K row vector **L** such that

$$\rho_\xi(x) = \mathbf{Lc} = \texttt{c2rMap}\,\mathbf{c}.$$

3. The composite mapping called `y2rMap` defined by

$$\texttt{y2rMap} = \rho_\xi(x) = \texttt{c2rMap}\,\texttt{y2cMap},$$

   which converts a data vector **y** directly into the probe value; this is a 1 by n row vector.

How is $\mathbf{L} = \texttt{c2rMap}$ actually calculated? In general, the computation includes the use of the all-important *inner product function* `inprod` to compute the integral (6.2). This function is working away behind the scenes in almost every functional data analysis. It evaluates the integral of the product of two functions (or the matrices defined by products of sets of functions), such as that defining the roughness penalty matrix $\mathbf{R} = \int L\phi L\phi'$ defined in Subsection 5.2.2. Where possible, this function uses an analytic expression for these integral values. However, more often than not, this computation requires numerical approximation.

The four important arguments to function `inprod` are as follows:

`fdobj1` Either a functional data object or a functional basis object.

`fdobj2` Also either a functional data object or a functional basis object. It is the integral of the products of these two objects that is computed. If either of these first two arguments are a basis object, it is converted to a functional data object with an identity matrix as its coefficient matrix.

`Lfdobj1` A linear differential operator object of class `Lfd` to be applied to `fdobj1`. If missing, the result of applying it is taken to be the function itself, that is, it is the *identity operator*.

`Lfdobj2` Also a linear differential operator object of class `Lfd` to be applied to `fdobj2`.

For the problem of computing `c2rMap`, one of the first two arguments would be a functional data object for the weight function ξ; the other would be the functional basis object used in the expansion of function x. As an illustration, consider a conventional linear regression model with design matrix $\mathbf{Z}$

$$\mathbf{y} = \mathbf{Z}\mathbf{c} + \mathbf{e},$$

where the regression coefficient vector $\mathbf{c}$ is estimated by ordinary least squares. Then, since $\mathbf{c} = (\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\mathbf{y}$, the matrix corresponding to `y2cMap` is $\mathbf{S} = (\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'$. Now suppose that for some reason we want to estimate the difference between the first and second regression coefficients, possibly because we conjecture that they may be equal in the population. Then the probe function ξ is equivalent to the probe vector $\mathbf{L} = (1,-1,0,\ldots)$, and this is the row vector corresponding to mapping `c2rMap`. Finally, the composite mapping `y2rMap` taking $\mathbf{y}$ directly into the value of this difference is simply the row vector $\mathbf{L}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'$.

For a more complicated example, suppose that we want to compare winter temperatures and precipitations for the 35 Canadian weather stations, and we have already defined basis objects `tempbasis` and `precbasis`, respectively. Suppose, too, that we have run the year from July 1 to June 30, so that winter is in the middle of the year. We can use as a probe function

$$\xi(t) = \exp\{20\cos[2 * \pi(t-197)/365]\},$$

which is proportional to the density for the von Mises distribution of data on a circle; the concentration parameter value 20 weights substantially about two months, and the location value 197 centers the weighting on approximately January 15 (see (Fisher et al., 1987) for more details.) The following code sets up the functional data object for ξ and then carries out the two integrations required for the two sets of 35 probe values produced by integrating the product of ξ with each of the basis functions in each of the two systems.

```
dayvec  = seq(0,365,len=101)
xivec   = exp(20*cos(2*pi*(dayvec-197)/365))
xibasis = create.bspline.basis(c(0,365),13)
xifd    = smooth.basis(dayvec, xivec, xibasis)$fd
tempLmat = inprod(tempbasis, xifd)
precLmat = inprod(precbasis, xifd)
```

The random behavior of the estimator of whatever we choose to estimate is ultimately tied to the random behavior of the data vector $\mathbf{y}$. Let us indicate the order n variance-covariance matrix of $\mathbf{y}$ as $\mathtt{Var}(y) = \Sigma_e$. Recall that we are operating in this chapter with the model

$$\mathbf{y} = x(\mathbf{t}) + \boldsymbol{\varepsilon} \ ,$$

where $x(\mathbf{t})$ here means the n-vector of values of x at the n argument values t_j. In this model $x(\mathbf{t})$ is regarded as fixed, and as a consequence $\Sigma_e = \mathtt{Var}(\boldsymbol{\varepsilon})$.

6.5.2 Computing Confidence Limits for Probe Values

We compute confidence limits in this book by a rather classic method: The covariance matrix Σ_ξ of $\xi = \mathbf{A}\mathbf{y}$ is

$$\Sigma_\xi = \mathbf{A}\Sigma_y\mathbf{A}'. \tag{6.3}$$

If the residuals from a smooth of the data have a variance-covariance matrix Σ_e, then we see from $\hat{\mathbf{c}} = \mathtt{y2cMap}\ \mathbf{y}$ that the coefficients will have a variance-covariance matrix

$$\Sigma_c = \mathtt{y2cMap}\ \Sigma_e\ \mathtt{y2cMap}'$$

We use the *conditional* variance of the residuals in this equation because we are only interested in the uncertainty in our estimate of $\mathbf{c}$ that comes from *unexplained* variation in $\mathbf{y}$ after we have explained what we can with our smoothing process. This in turn estimates the random variability in our estimate of the smooth.

We apply (6.3) a second time to get the variance-covariance matrix Σ_ξ for a functional probe by

$$\Sigma_\xi = \mathtt{c2rMap}\ \Sigma_c\ \mathtt{c2rMap}' = \mathtt{c2rMap}\ \mathtt{y2cMap}\ \Sigma_e\ \mathtt{y2cMap}'\ \mathtt{c2rMap}'. \tag{6.4}$$

6.5.3 Confidence Limits for Prince Rupert's Log Precipitation

We can now plot the smooth of the precipitation data for Prince Rupert, British Columbia, Canada's rainiest weather station. The log precipitation data are stored in 365 by 35 matrix `logprecav`, and Prince Rupert is the 29th weather station in our database. We first smooth the data:

```
lambda    = 1e6;
fdParobj  = fdPar(daybasis, harmaccelLfd, lambda)
logprecList= smooth.basis(day.5, logprecav, fdParobj)
logprec.fd = logprecList$fd
fdnames = list("Day (July 1 to June 30)",
           "Weather Station" = CanadianWeather$place,
           "Log 10 Precipitation (mm)")
logprec.fd$fdnames = fdnames
```

Next we estimate Σ_e, which we assume is diagonal. Consequently, we need only estimate the variance of the residuals across weather stations for each day. We do

this by smoothing the log of the mean square residuals and then exponentiating the result:

```
logprecmat  = eval.fd(day.5, logprec.fd)
logprecres  = logprecav - logprecmat
logprecvar  = apply(logprecres^2, 1, sum)/(35-1)
lambda      = 1e8
resfdParobj = fdPar(daybasis, harmaccelLfd, lambda)
logvar.fit  = smooth.basis(day.5, log(logprecvar),
                           resfdParobj)
logvar.fd   = logvar.fit$fd
varvec      = exp(eval.fd(daytime, logvar.fd))
SigmaE      = diag(as.vector(varvec))
```

Next we get `y2cMap` from the output of `smooth.basis`, and compute `c2rMap` by evaluating the smoothing basis at the sampling points. We then compute the variance-covariance matrix for curve values, and finish by plotting the log precipitation curve for Prince Rupert along with this curve plus and minus two standard errors. The result is Figure 6.6.

```
y2cMap = logprecList$y2cMap
c2rMap = eval.basis(day.5, daybasis)
Sigmayhat = c2rMap %*% y2cMap %*% SigmaE %*%
          t(y2cMap) %*% t(c2rMap)
logprec.stderr = sqrt(diag(Sigmayhat))
logprec29 = eval.fd(day.5, logprec.fd[29])
plot(logprec.fd[29], lwd=2, ylim=c(0.2, 1.3))
lines(day.5, logprec29 + 2*logprec.stderr,
        lty=2, lwd=2)
lines(day.5, logprec29 - 2*logprec.stderr,
        lty=2, lwd=2)
points(day.5, logprecav[,29]))
```

6.6 Some Things to Try

1. The 35 Canadian weather stations are divided into four climate zones. These are given in the character vector `CanadianWeather$region` that is available in the `fda` package. After computing and plotting the variance-covariance functional data object for the temperature data, compare this with the same analysis applied only to the stations within each region to see if the variability varies between regions. In Chapter 10 we will examine how the mean temperature curves changes from one region to another.
2. What does the covariance bivariate functional data object look like describing the covariation between temperature and log precipitation?

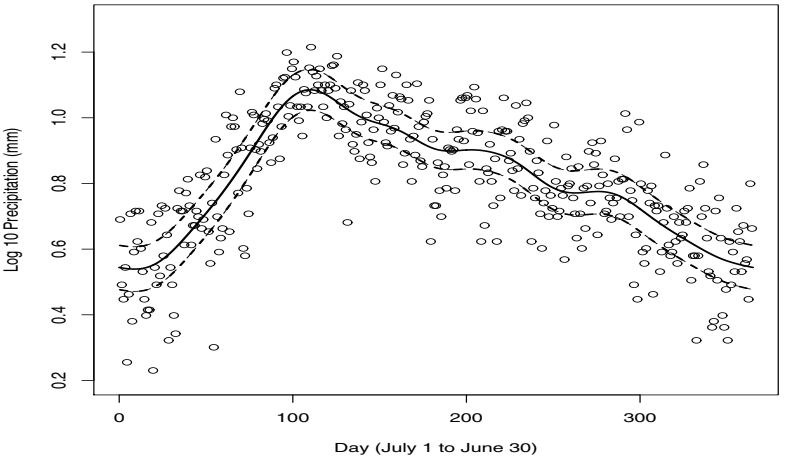

Fig. 6.6 The solid curve is the smoothed base 10 logarithm of the precipitation at Prince Rupert, British Columbia. The dashed lines indicate 95% pointwise confidence limits for the smooth curve based on the data shown as circles.

3. Examine the phase-plane diagram for each of the temperature curves.
4. Compute the standard deviation function for the precipitation data and for the log precipitation data. For each case, plot values of the standard deviation function against values of the mean function. Do you see a general linear trend for the precipitation data and less of that trend for the log precipitation data?
5. Examine the residuals for the growth data from their monotone smooths. Do they appear to be normally distributed or do they exhibit long tails? Do the error variances seem to vary substantially from child to child? Are there any outliers, perhaps due to a failure of the smoothing algorithm, or problems with the measurement process? How does error variance depend on age?
6. Explore the residuals for correlation structure. How would one do this when the data are not equally distributed? One possibility is to treat them as spatial data, and use methods developed in that domain to answer these questions.