15

Analysis of Spatiotemporal Data

15.1 Introduction

Many ecological data sets include not only a spatial but also a temporal component. Data may be collected in an agricultural field trial over a period of years in the same field. An uncultivated ecosystem may be monitored over an extended period of time. Both of these processes lead to spatiotemporal data sets. In this chapter, we discuss methods for dealing with data that contain both spatial and temporal components. The theory for spatiotemporal data is much less well developed than that for dealing with purely spatial or purely temporal data, and the contents of this chapter are somewhat *ad hoc* in nature. Perhaps unsurprisingly, this is also one of the most active areas of research, and one of the most dynamic, both in the development of the theory and in the development of analytical methods and software. The development of the theory took a major step forward with the publication of the book by Cressie and Wikle (2011). As Gräler et al. (2016) point out, however, the implementation of the theory has lagged behind. For this reason, the chapter consists of a diverse selection of methods that attempt to put some of the material discussed by Cressie and Wikle (2011) into the context of the data used in this book. Section 15.2 deals with spatiotemporal variograms and kriging, that is, the interpolation of spatial quantities in both space and time. Section 15.3 introduces the concept of spatiotemporal process models derived from physicochemical models for diffusion-reaction processes. Section 15.4 introduces discrete time approximations to spatiotemporal processes, sometimes called "state and transition models." Finally, Section 15.5 describes a Bayesian approach to the analysis of spatiotemporal processes.

Because spatiotemporal data analysis is such a dynamic area, both in development of the theory and in development of the R code, it is not unlikely that some aspects of R for analysis of spatiotemporal data may have changed by the time you are reading this. If you run into problems, use the resources discussed in Section 2.7 to help you solve them.

15.2 Spatiotemporal Data Interpolation

15.2.1 Representing Spatiotemporal Data

Figure 15.1 shows a set of spatiotemporal data that we will be analyzing in this section. The data set is four years of yield monitor data from Field 4.1. The crops grown in the four successive years were wheat, tomato, bean, and sunflower. The open circles

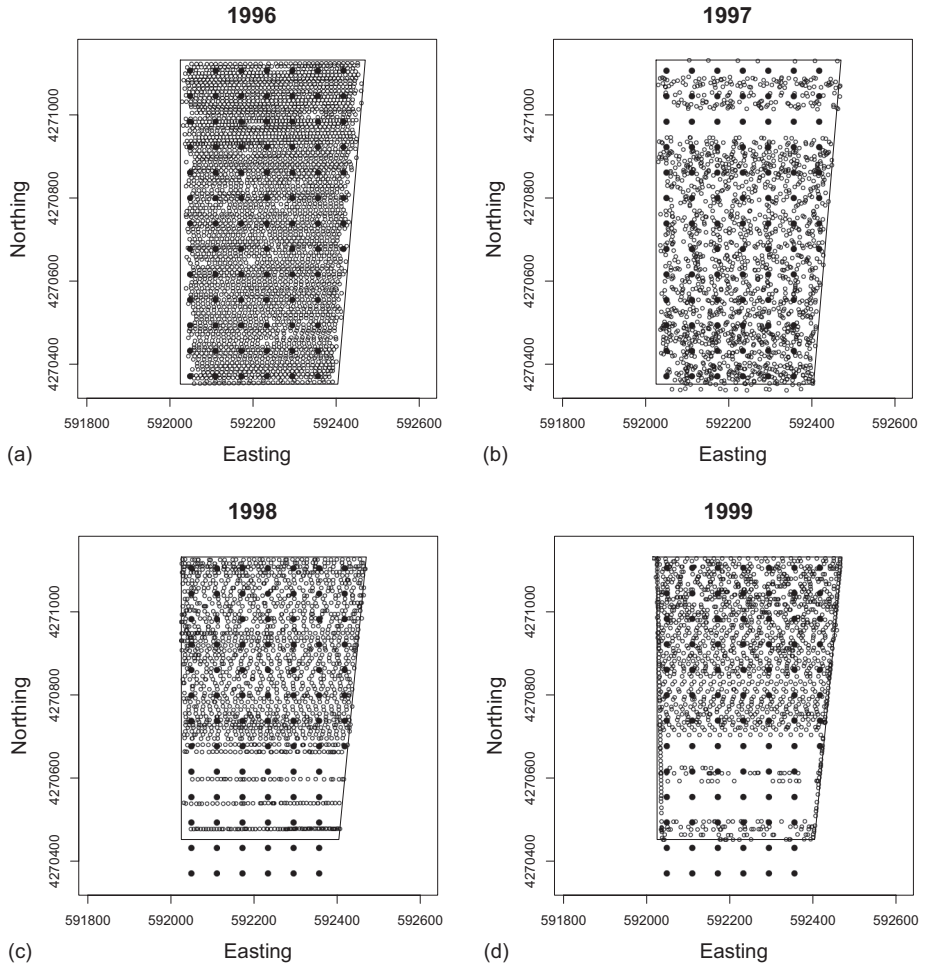

FIGURE 15.1
Yield monitor data from four years of harvests in Field 1 of Data Set 4. The location of every tenth data record is shown. The black dots indicate the hand-sampling locations. Yields from (a) 1996; (b) 1997; (c) 1998; (d) 1999.

represent yield data measured in each year, and the dark circles represent locations where manually collected data were recorded. Although the dark circles are shown in each figure, these data were only recorded in the first year. So far we have only analyzed the data from the first year, 1996. A glance at the figure shows some of the issues that arise with spatiotemporal data sets. The figure also illustrates some of the difficulties of field research (difficulties that are sometimes not fully appreciated by our laboratory-based colleagues). The first problem revealed in the figure is that the southern portion of the field was removed from the experiment after two years. The reason is this. One of the principles of precision agriculture is that spatially precise information should be useful to the farmer in increasing profitability by permitting a greater efficiency in resource use. The cooperator who farms the fields in Data Set 4 is very astute and a very good farmer, and he put this principle into practice in our study. Although he knew that the south end of Field 4.1 had better soil than the rest of the field, he did not know exactly where this good soil was located; nor did he know how good it was relative to the rest

of the field. After seeing the results of our first two years of the experiment, he made a decision that was good for his profitability but bad for the experiment: he took the southern part of the field out of field crop production and put it into a high-value crop (seed onions). Therefore, the analysis of the four years of data must exclude the southernmost 12 data locations.

The second problem is that yield monitor coverage was incomplete in each of the years after the first. This was partly due to equipment problems and partly due to the need of the cooperator to harvest the material in a timely manner. In 1998, when not all the field could be surveyed, yield monitor passes were made along a line that passed close to the sample points, as shown in Figure 15.1c. In 1997 and 1999, however, there are sample points with no yield data near to them. The third apparent problem really isn't one. The GPS located on the yield monitor in 1997 appears to have been in error in the southern part of the field, since some of the recorded data locations are apparently outside the boundary of the field (Figure 15.1b). This is actually not a problem. The boundary polygon was adjusted in this book for expository purposes and is actually slightly smaller than the field itself.

In this section we are going to put the data into objects of the class `spacetime`, from the R package of the same name (Pebesma, 2011, 2012). This has rapidly become the standard format for spatiotemporal data, particularly for purposes of interpolation. To prepare the data for analysis, we first follow the procedure used in Section 6.3.1 to interpolate the 1997, 1998, and 1999 yield data to the 1996 sample points using inverse distance weighted interpolation. Since we did not eliminate the last two rows of the sample points from 1996, the 1998 and 1999 data include records for data that do not really exist. We therefore eliminate these data records from each year. To keep the code concise, we use much shorter names for the variables than we have in the previous chapters.

```
> Y96 <- data.Set4.1.96[(1:74),]
> Y97 <- data.Set4.1.97[(1:74),]
> Y98 <- data.Set4.1.98[(1:74),]
> Y99 <- data.Set4.1.99[(1:74),]
```

We are going to put the data into an object of class `STFDF`. This extends the `sp` classes for points, lines, and polygons to include a time component. The class is one of those developed in the `spacetime` package. First, we create the attribute data values for the `STFDF` object. Because the absolute magnitudes of the yields for the four different crops are very different, we normalize each year's yield to a scale of 0 to one 100, creating a of normalized values for each year.

```
> Y96.norm <- 100 * Y96[,2] / max(Y96[,2])
> Y97.norm <- 100 * Y97[,2] / max(Y97[,2])
> Y98.norm <- 100 * Y98[,2] / max(Y98[,2])
> Y99.norm <- 100 * Y99[,2] / max(Y99[,2])
```

The attribute values are stored as a data frame with a single data field, so we concatenate them to create the data set we will use.

```
> Yield.norm <- data.frame("Yield" = c(Y96.norm,
+ Y97.norm, Y98.norm, Y99.norm))
> nrow(Yield.norm)
[1] 296
```

The spatial portion of a `spacetime` object is stored as an `sp` object called `Yield.sites`, and this is created using the function `coordinates()` in the usual way.

```
> Yield.sites <- data.frame(Easting = Y96$Easting, Northing =
+    Y96$Northing)
> coordinates(Yield.sites) <- c("Easting", "Northing")
> proj4string(Yield.sites) <- CRS("+proj=utm +zone=10 +ellps=WGS84")
```

There are 74 position coordinates and a data frame four times that long of attribute values, so the `spacetime` object will automatically recycle through the position coordinates to obtain the correct location for each attribute value.

We have not yet had to represent time sequences in this book. R treats temporal reference systems in a way analogous to how it treats spatial coordinate reference systems, which are briefly discussed in Section 2.4.3. Specifically, with spatial coordinates there are a variety of packages and functions such as `st_crs()` and `proj4string()` that have been created by contributors to deal with standard coordinate reference systems such as longitude-latitude and UTM. Similarly, there are base and contributed packages in R that allow the user to deal with standard temporal reference systems. One of these is `ISOdate()`, which is included in the base package. We will use this to create the time data.

```
> Yield.year <- c(1996, 1997, 1998, 1999)
> Yield.month <- c(5, 9, 9, 9)
> Yield.day <- rep(15, length(Yield.year))
> print(Yield.time <- ISOdate(year = Yield.year, month =
+    Yield.month, day = Yield.day))
[1] "1996-05-15 12:00:00 GMT" "1997-09-15 12:00:00 GMT"
[3] "1998-09-15 12:00:00 GMT" "1999-09-15 12:00:00 GMT"
```

The values are represented in the International Standards Organization (ISO) format for year, month, day, hour, minute, and second. In our case, we just approximate the harvest dates, which were in the spring for the 1996 wheat crop and in the fall for the other crops. The `spacetime` package makes extensive use of the contributed R package `zoo`, described by Zeileis and Grothendieck (2005), to handle temporal data, and this reference provides a good discussion of the representation of temporal data.

We are now ready to create an `STFDF` object. In order to be able to plot it nicely, we will create one with polygon data. Here we use the very convenient property of `sp` objects that they can for many operations be treated as data frames. This allows us to simply drop the lowest two rows of polygons from the data frame.

```
> thsn.sf <- st_read("auxiliary\\set4.1thiessen.shp")
> thsn.sp <- as(thsn.sf, "Spatial")
> proj4string(thsn.sp) <- CRS("+proj=utm +zone=10 +ellps=WGS84")
> thsn9899.sp <- thsn.sp[1:74,]
```

Now we create the polygon object. We can use the fact that the `spacetime` plotting function `stplot()` works through the `sp` function `spplot()`, which in turn works through the `lattice` package (Section 2.6.3). Figure 15.2 shows the result.

```
> YieldPolys.stfdf <- STFDF(thsn9899.sp, Yield.time,
+    data = Yield.norm)
> library(lattice)
```

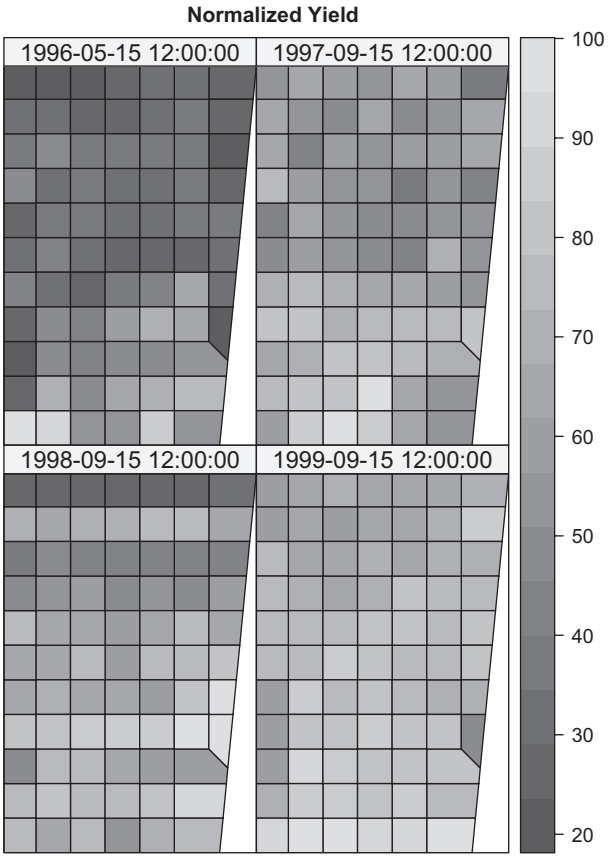

FIGURE 15.2
Plot of the data space of normalized yields in Field 4.1 in 1996 and 1997 showing the four clusters obtained via *k*-means clustering with $k = 4$.

```
> greys <- grey(seq(5, 19) / 22)
> trellis.device(color = FALSE)
> stplot(YieldPolys.stfdf, col.regions = greys, main = "Normalized
+ Yield")
```

Recall that the crop sequence was wheat, tomato, bean, and sunflower. Wheat is the only winter crop, and clearly suffered more from the heavy soil in the north.

Now that we have assembled the spatiotemporal data, in the next two sections we will discuss one important application: the spatiotemporal variogram and its application to spatiotemporal kriging. For purposes of comparison in this section, we fit four individual variograms to the sequence of four years of normalized yield data. Figure 15.3a shows a perspective plot created using the function `persp()` of the four empirical variograms. Figure 15.3b shows the same plot, but with the `dist` variable replaced by `cutoff - dist`, where `cutoff` is the cutoff distance of the variogram, 400 m. This figure is shown to provide ease of comparison the figures that will be shown in the next section. These are created using the `lattice` function `wireframe()`, which arranges the axes differently from that of `persp()`. After 1996, the individual variograms tend to be mostly nugget, indicating little spatial autocorrelation and even less temporal autocorrelation.

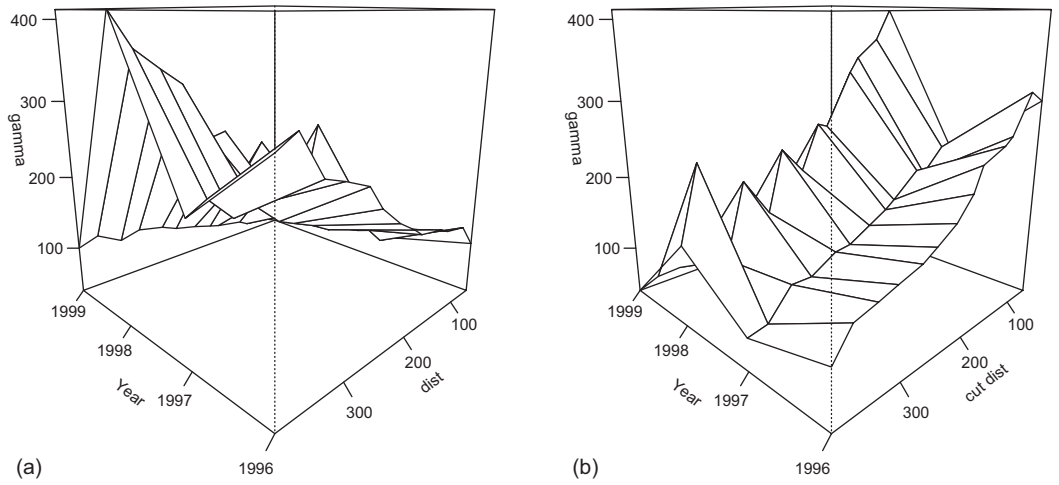

FIGURE 15.3
(a) Perspective plot of four empirical variograms of yield in Field 4.1. (b) Same variogram rotated 180°.

15.2.2 The Spatiotemporal Variogram

In Section 6.3.2, we saw that kriging interpolation is carried out using the covariogram. Recall (Equation 4.20) that for stationary, isotropic spatial data defined on a domain D (Section 1.2.1) the *variogram* is defined as

$$\gamma(h) = \frac{1}{2}\text{var}\{Y(x+h) - Y(x)\}, \tag{15.1}$$

where Y is the measured quantity and x is a position vector with coordinates (x, y). The quantity h is the spatial lag. In Section 4.6.2, we noted that for a purely spatial process if we define the covariogram as (Equation 4.24)

$$C(h) = \text{cov}\{Y(x), Y(x+h)\}, \tag{15.2}$$

then from Equation 4.26 the covariogram is related to the variogram by

$$C(h) = C(0) - \gamma(h), \tag{15.3}$$

where $C(0)$ is the sill of the variogram. Consider now a spatiotemporal domain, which we will denote $D \times T$ with elements (x_i, t_j). Let u represent a temporal lag analogous to the spatial lag h. Then we can write the spatiotemporal variogram as (Cressie and Wikle, 2011, p. 315)

$$\gamma(h,u) = \frac{1}{2}\text{var}\{Y(x+h, t+u) - Y(x,t)\}. \tag{15.4}$$

Similarly, if we define the spatiotemporal covariogram as

$$C(h,u) = \text{cov}\{Y(x+h,t+u) - Y(x,t)\}, \tag{15.5}$$

then

$$C(h,u) = C(0,0) - \gamma(h,u). \tag{15.6}$$

This leads to a complication in estimating the variogram. The spatial dimension is assumed stationary and isotropic (Section 3.2.2), which permits to express the covariogram in the simple form of Equation 15.2. We cannot, however, extend this spatial isotropy assumption to the covariance function $C(h, u)$ defined in Equation 15.5 because although the process is stationary in time, it is not isotropic between the time dimension and the spatial dimension.

In computing a purely spatial empirical variogram in Section 4.6.2, we used a method of moments estimator (the formula is given by Equation 4.21). In the case of the spatiotemporal variogram, it is still possible to compute a simple estimator of this form. This simple estimator may not, however, be consistent with the spatiotemporal covariance of the data. It is, therefore, often desirable to make a particular assumption about the relationship between the variogram in the spatial dimension and that in the temporal dimension, that is, in the form of the covariogram function $C(h, u)$ in Equation 15.6. We will present a few of the options discussed by Gräler et al. (2016). Our discussion is based on code written by Gräler (2016) and by Veronesi (2015).

We start with the same four yield maps used in the previous section. If we are going to compute a variogram, however, we are going to need more than four points in the time direction. Webster and Oliver (1992) suggest at least 100 points for a spatial variogram. We will, therefore, create some fake data that gives us at each of the 74 spatial coordinates a sequence of 100 temporal values. To create the 100 temporal values, we will for each of the 74 locations first fit the 4 normalized year values with a cubic spline (see Section 9.2) denoted s and then generate a sequence of 100 values Y_i according to the equation

$$\begin{aligned} Y_i &= s_i + \eta_i, \\ \eta_i &= \lambda\eta_{i-1} + \varepsilon_i. \end{aligned} \tag{15.7}$$

This is a first-order autoregressive time series of the type discussed in Section 3.5.1, and provides some degree of temporal autocorrelation to the data. Here η_l are the autocorrelated random variables, λ is a constant whose value is between 0 and 1, and the ε_i are independent, normally distributed random variables with mean 0 and variance σ^2. The code is very straightforward and is not shown. Figure 15.4 shows for the first spatial data location the four yield values and the full temporal data set, with the four "real" data values shown as black dots and the 96 artificial data values shown as open circles. The dashed line shows the spline fit, and the solid line connects the 100 data values. To remove any connection with yield and years for this artificial data set we simply call the data values $Y(x, t)$. The temporal autocorrelation is apparent. The data are stored in a data frame.

```
> Y.data <- data.frame(Y = Y)
> nrow(Y.data)
[1] 7400
```

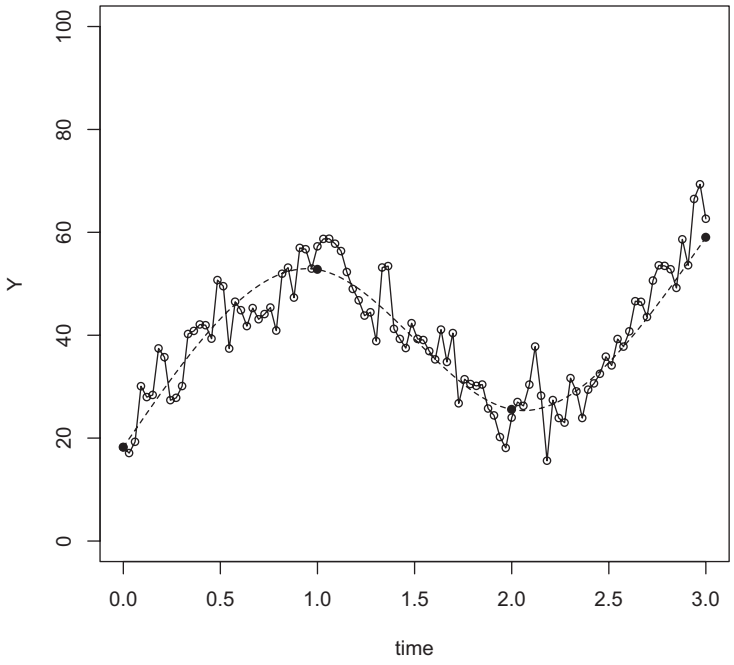

FIGURE 15.4
Artificial data set created by fitting a first-order autoregressive model to the time series of four years of yield data in Field 4.1. The yield values at the first data location are shown; the other 73 set of values are similar.

The location data object is created in the same way as in Section 15.2.1 except that in this case we create point rather than polygon data.

```
> Y.sites <- data.frame(Easting = Y96$Easting, Northing = Y96$Northing)
> coordinates(Y.sites) <- c("Easting", "Northing")
> proj4string(Y.sites) <- CRS("+proj=utm +zone=10 +ellps=WGS84")
```

In creating the temporal data, we need to be a bit careful. The functions that we will be using to create the spatiotemporal variogram work better if the time values are evenly spaced (this property is called being *strictly regular*). If we use the four successive years 1996 through 1999 as the time values, then because 1996 is a leap year the data will not be strictly regular. Since the time variable, like the attribute data, no longer has any physical meaning, we will simply use the function `as.date()` to create a sequence of 100 days as members of the date class. This class numbers days sequentially beginning with January 1, 1960 as day 0.

```
> as.date(0:3)
[1] 1Jan60 2Jan60 3Jan60 4Jan60
> class(as.date(0:3))
[1] "date"
```

Therefore, we set a time of the first 100 days in 1960. R has two basic time classes, `POSIXct` and `POSXlt`. The class `POSIXct` represents calendar days beginning with January 1, 1970 as day 0. We can use a coercion function to covert our date values to `POSIXct` values.

```
> as. POSIXct(as.date(0:3))
[1] "1959-12-31 16:00:00 PST" "1960-01-01 16:00:00 PST"
[3] "1960-01-02 16:00:00 PST" "1960-01-03 16:00:00 PST"
```

Note that because I am writing this in California and my computer is therefore set to California time, `as.POSITct()` automatically converts GMT to Pacific Standard Time. This doesn't matter in our case since it is all fake data anyway. We can now create the temporal sequence and the `STFDF` object.

```
> Y.time <- as.POSIXct(as.date(0:(nt - 1)))
> Y.stfdf <- STFDF(Y.sites, Y.time, data = Y.data)
```

We are now ready to begin trying various forms of spatiotemporal variogram estimation. We begin with the simple empirical variogram, making no assumptions about the form of the covariogram $C(h, u)$. Here is the code.

```
> Y.empvgm <- variogramST(Y ~ 1, data = Y.stfdf, tlags = 0:7,
+   cutoff = 400, width = 100)
```

The first two arguments are obvious. The argument `tlags` specifies the number of time lags at with the variogram will be estimated, `cutoff` specifies the spatial separation distance up to which point pairs are included in semivariance estimates, and `width` specifies the width of subsequent distance intervals into which data point pairs are grouped for semivariance estimates.

Figure 15.5 shows a wireframe plot of the empirical variogram. Comparing this figure with Figure 15.3b (note that the time axes in the figures move in opposite directions), we see that the empirical spatiotemporal variogram is smoother than the corresponding set of independent variograms, presumably reflecting the temporal autocorrelation of the former. Once again, however, the variograms after the initial time are mostly nugget.

Now we turn to constructing variograms reflecting models for the structure of the covariogram $C(h, u)$. Gräler et al. (2016) describe five models of increasing complexity. We will discuss two of them. The simplest is the *separable* model, which assumes a relationship of the form

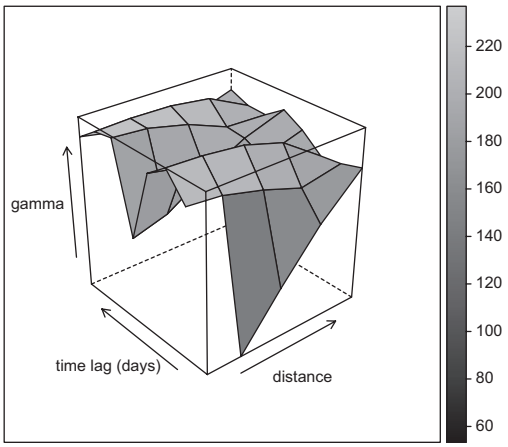

FIGURE 15.5
Wireframe plot of the spatiotemporal variogram of the artificial data set.

$$C(h,u) = C_s(h)C_t(u). \tag{15.8}$$

The empirical variogram assuming a variogram model is computed using the `gstat` function `vgmST()` and then fit using the function `fit.Stvariogram()`. Both of these functions have very many arguments, and I am not going to show the code, which can be viewed in the accompanying R code file. Veronesi (2015) discusses these arguments to some extent. They are mostly control parameters and, as he points out, they are mostly selected by trial and error. In the examples of this section, I did not find the results to be strongly dependent on the values of the control parameters.

Figure 15.6 shows a wireframe plot of the fitted surface. The model is practically pure nugget after the first year. The process of selecting the best of the candidate models in an actual fitting process would ordinarily involve minimizing the mean square error. This is provided as an attribute of the `StVariogramModel` object.

```
> attr(Y.sep.fit, "MSE")
[1] 324.2339
```

Veronesi (2015), who analyzed a real data set, found that his separable model also yielded very poor fit.

The sum-metric model assumes a relationship of the form

$$C(h,u) = C_s(h) + C_t(u) + C_j(\sqrt{h^2 + \kappa u^2}), \tag{15.9}$$

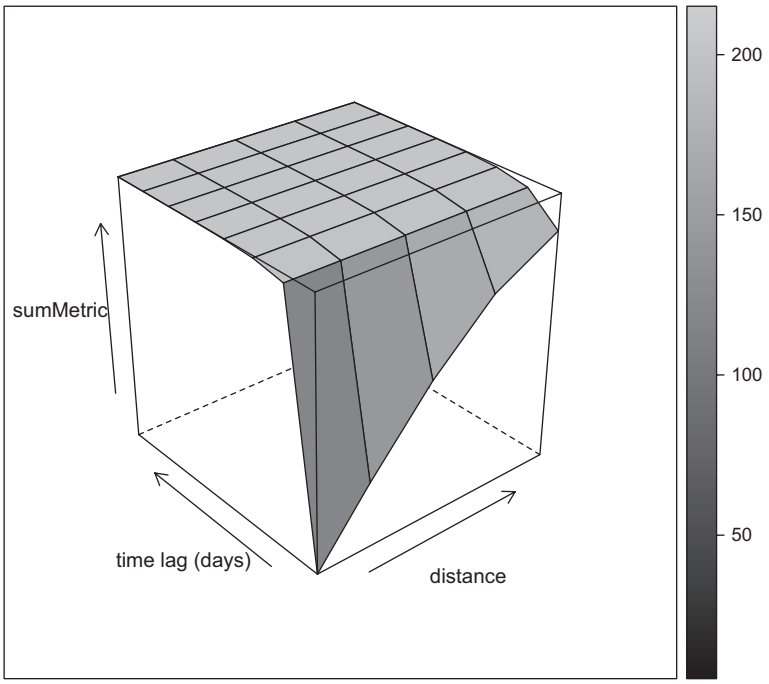

FIGURE 15.6
Wireframe plot of the spatiotemporal variogram of the artificial data set fitter using a separable model given by Equation 15.8.

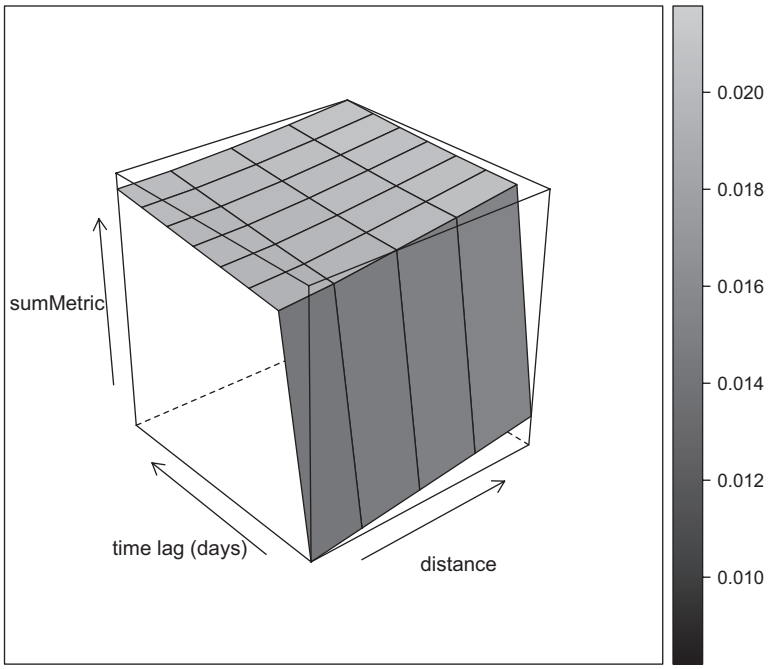

FIGURE 15.7
Wireframe plot of the spatiotemporal variogram of the artificial data set fitter using a sum-metric model given by Equation 15.9.

where κ is a spatial anisotropy parameter whose value can be obtained using the function `stAni()`. Figure 15.7 shows the resulting wireframe plot. It again almost pure nugget after the first time step. We can see that the sum-metric model provides a slightly better fit.

```
> attr(Y.smm.fit, "MSE")
[1] 310.3402
```

One application of the spatiotemporal variograms, as with the purely spatial variogram, is the visualization of spatial relationships. In that context, the empirical spatiotemporal variogram provides an indication that after the first year the spatiotemporal variogram is mostly pure nugget, meaning that the data are not highly spatially autocorrelated. As with the pure variogram, probably the most important use of the spatiotemporal variogram model is in its application to kriging interpolation. This is discussed in the next section.

15.2.3 Interpolating Spatiotemporal Data

Recall (Equation 6.2) that an interpolator $\hat{Y}(x)$ is called a *linear interpolator* if for some set of coefficients ϕ_i, $i = 1, 2, \ldots, n$,

$$\hat{Y}(x) = \sum_{i=1}^{n} \phi_i Y(x_i). \tag{15.10}$$

An interpolator $\hat{Y}(x)$ is *unbiased* if the interpolation functions ϕ_i satisfy

$$\sum_{i=1}^{n} \phi_i = 1. \tag{15.11}$$

The important property of the kriging interpolator is that it is the best linear unbiased estimator in the sense that is minimizes the error variance. The same conditions apply for a spatiotemporal linear interpolator. The equations are exactly the same, except that now Y and $\hat{Y}$ are functions of x and t, so we have

$$\hat{Y}(x,t) = \sum_{i=1}^{n} \varphi_i Y(x_i, t_j). \tag{15.12}$$

The package `gstat` provides a wrapper function `krigeST()` that, once a spatiotemporal variogram model is developed, makes spatiotemporal kriging very easy. We will continue our discussion with the example begun in Section 15.2.2.

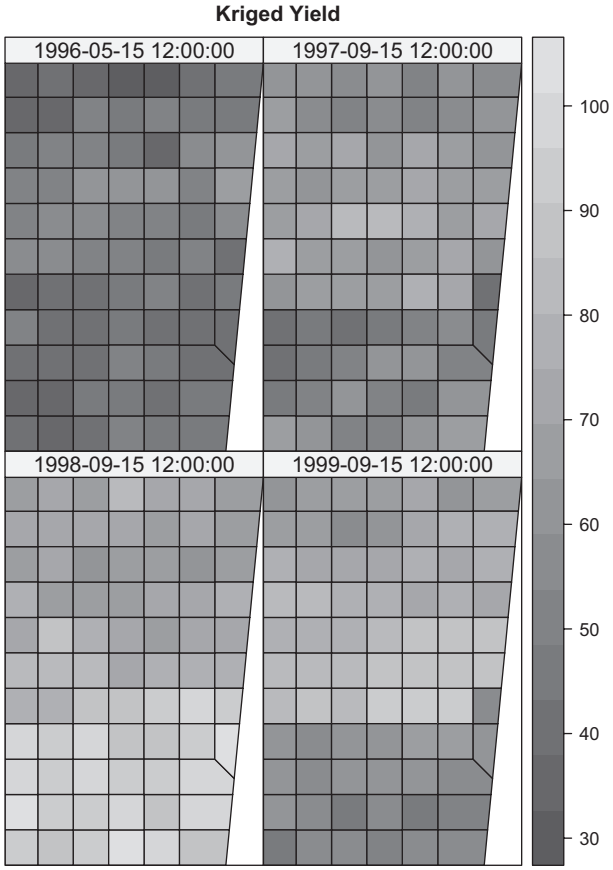

FIGURE 15.8
Thiessen polygons showing kriged spatiotemporal "yield" values.

We will use the sum-metric variogram model computed in the previous section in our interpolation. This variogram is in the object `Y.fit.smm`. In order to compare the interpolated values with the original in as simple a manner as possible, we will plot the kriged values at the times corresponding to the four original harvests.

```
> Y.predtime <- as. POSIXct(as.date(seq(24.5,nt - 0.5, 25)))
```

We then use the code from Section 15.2.1 to enter the year values 1996 through 1999 as the temporal data. The kriging is carried out using a call to the function `krigeST()`.

```
> Y.pred <- krigeST(Y ~ 1, data = Y.stfdf, modelList = Y.fit.smm,
+   newdata = STF(Y.sites, Y.predtime))
```

Figure 15.8 shows the results, plotted as Thiessen polygons. As with many interpolations, the interpolated values are quite different from the original values in Figure 15.2. In particular, the lower yields in the northern end in 1996 and 1997 have been shifted toward the southern end in 1997 and 1999. The results of this section do not lead to any further biophysical insights about the factors influencing crop yield in Field 4.1, and should only be considered as an introduction to how spatiotemporal kriging is carried out.

15.3 Spatiotemporal Process Models

15.3.1 Models for Dispersing Populations

Cressie and Wikle (2011, Chapter 6) recommend the use of process models to characterize spatiotemporal processes. The model that they find generally most appropriate is the diffusion-reaction model, and that is the subject of this section. We begin by discussing a model for diffusion. Consider a petri dish with a very shallow layer of water, and suppose a drop ink is released into the water. If $Y(x, y, t)$ describes the concentration of ink at location (x, y) at time t, then Y can be modeled using the *diffusion equation*. This equation is written

$$\frac{\partial Y}{\partial t} = D\left(\frac{\partial^2 Y}{\partial x^2} + \frac{\partial^2 Y}{\partial y^2}\right), \tag{15.13}$$

together with appropriate boundary conditions (Dettman, 1969, p. 145). We will assume that the equation is defined on an infinite plane with zero flux in the limit. Here D is the diffusion coefficient and measures the rate at which the ink disperses into the water. The diffusion equation also has a long history modeling the movement of species in the environment (Okubo, 1980). If $Y(x, y)$ represents a population that is entirely concentrated at the origin $(x, y) = (0, 0)$ at time $t = 0$ with a population Y_0, and if there is no mortality, then for any $t > 0$ the population $Y(x, y, t)$ will be described by the solution of Equation 15.13 on an infinite plane. This solution is a normal density (Plant and Cunningham, 1991),

$$Y(x, y, t) = \frac{Y_0}{4\pi Dt}\exp\left(\frac{-(x^2 + y^2)}{4Dt}\right). \tag{15.14}$$

If mortality is included in the population model, then Equation 15.13 is modified to

$$\frac{\partial Y}{\partial t} = D\left(\frac{\partial^2 Y}{\partial x^2} + \frac{\partial^2 Y}{\partial y^2}\right) - \mu Y, \tag{15.15}$$

where μ is the mortality rate (assumed fixed), and Equation 15.14 becomes

$$Y(x, y, t) = \frac{Y_0 e^{-\mu t}}{4\pi D t} \exp\left(\frac{-(x^2 + y^2)}{4Dt}\right). \tag{15.16}$$

This model assumes that the organisms behave as passively diffusing entities, with no directed behavior, and that the entire population is concentrated at a single point at time $t = 0$. More complex initial conditions are relatively easy to take care of, but the issue of behavior may not be.

Plant and Cunningham (1991) developed a model of the form of Equation 15.15 to describe the results of an experiment in which marked sterile Mediterranean fruit flies (medflies, *Ceratitis capitata* W.) were released from a single location in a macadamia orchard on the island of Hawaii. Traps using a pheromone bait were arranged in a regular grid throughout the orchard. Because medflies are fairly weak fliers, the attractive properties of the bait were not thought to seriously disrupt the spatial distribution of the population.

This biophysical system is one that should be amenable to a diffusion model if any such system is. Insects, particularly weak fliers like the medfly, may not display substantial nonrandom dispersal behavior, and the orchard provided a relatively uniform environment. Plant and Cunningham found that it was necessary to modify the model of Equation 15.15 to include a convection term that accounted for a prevailing wind. In addition, to obtain a good fit it was necessary to divide the population into a mobile and a "settled" subpopulation, with the latter no longer changing position. In the model, flies moved from the mobile to the settled population at a fixed rate. The model as modified was fit to the data using a nonlinear least squares algorithm (see Exercise 12.5), and found to provide a reasonable fit.

Returning to the analogy of ink dispersing in a petri dish, the "mortality" term μY may be thought of as modeling a chemical reaction in which the ink is gradually transformed into some other chemical, so that its concentration declines. From this interpretation, equations such as Equation 15.15 are often called *diffusion-reaction equations* (Winfree, 1977). It is also possible to include more complex "reaction" functions $f(Y)$.

Of course the annual crop yields that are the subject of Section 15.2 do not in any sense "diffuse" or "react." Many spatiotemporal processes in ecology have do not have biophysical properties from which a diffusion equation could be derived from first principles. The interesting question is whether a diffusion-reaction model that incorporates an error term can provide a purely empirical description of a spatiotemporal process, in the same way that a linear regression model does not have to imply any causal relationship between the explanatory and response variables but only an empirically observed relationship. In the next section, we will explore the application of this approach to our yield data.

15.3.2 A Process Model for the Yield Data

The yield process represented in Figure 15.1 is continuous in space (approximately) and discrete in time. The data, however, are discrete in both space and time. Although we will

represent the "diffusion-reaction" model for this process abstractly using a continuous spatiotemporal model such as Equation 15.15 we will implement this model using discrete space and time. Our approach is motivated by the methods used in the solution of ordinary and partial differential equations. We begin by considering an ordinary differential equation of the form

$$\begin{aligned} \frac{dY}{dt} &= f(Y), \\ Y(0) &= Y_0. \end{aligned} \tag{15.17}$$

The simplest method to compute a numerical solution to this equation is called *Euler's method* (Press et al., 1986, p. 543). We approximate the derivative on the left-hand side of the first equation by a forward difference operator

$$\delta_t \cong \frac{Y(t+\Delta t) - Y(t)}{\Delta t}. \tag{15.18}$$

This allows us to write symbolically

$$\begin{aligned} \delta_t Y &= f(Y), \\ Y(0) &= Y_0 \end{aligned} \tag{15.19}$$

and to implement this numerically as

$$Y(t+\Delta t) \cong Y(t) + \Delta t Y(t). \tag{15.20}$$

Starting with $Y(0) = Y_0$, we can march forward in time. There are of course much more sophisticated numerical methods for the solution of ordinary differential equations, but we will base our representation of the temporal part of the discrete "diffusion-reaction" process on Equation 15.20.

The spatial part of the model is based on the so-called second centered difference. We write

$$\frac{\partial^2 Y}{\partial x^2} \cong \frac{\frac{\partial Y}{\partial X}(x+\Delta x, y, t) - \frac{\partial Y}{\partial X}(x-\Delta x, y, t)}{2\Delta x}. \tag{15.21}$$

The first partial derivative in the numerator of Equation 15.21 may be approximated as a forward difference

$$\frac{\partial Y}{\partial X}(Y(x+\Delta x, y, t) \cong \frac{Y(x+\Delta x, y, t) - Y(x, y, t)}{\Delta x}, \tag{15.22}$$

and the second partial derivative may be approximated by the corresponding backward difference. Plugging these into Equation 15.21 allows us to write

$$\frac{\partial^2 Y}{\partial x^2} \cong \frac{Y(x+\Delta x, y, t) - 2Y(x, y, t) + Y(x-\Delta x, y, t)}{2\Delta x^2}, \tag{15.23}$$

and we define the second centered difference δ_{xx}^2 to be the quantity on the right-hand side. The second centered difference in the y direction is defined similarly. This allows us to write a discrete spatiotemporal analog to Equation 15.15 as

$$\delta_t Y = D\left(\delta_{xx}^2 Y + \delta_{yy}^2 Y\right) + \mu Y + \varepsilon_{xyt}. \tag{15.24}$$

where an error term ε_{xyt} is explicitly incorporated. In the numerical solution of partial differential equations, Equation 15.24, without the error term, is a very crude method called an explicit method (Ames, 1977, p. 62), but we are interested in testing it as the basis for an empirical model of a spatiotemporal process. We have changed the sign of the coefficient μ because this term no longer represents mortality or any other biophysical quantity; Equation 15.24 is purely an empirical fit to data.

Expanding δ_t, δ_{xx}, and δ_{yy} yields after a little algebra

$$\begin{aligned} Y(t+\Delta t,x,y) = Y(t,x,y) + D\left(\frac{Y(t,x+\Delta x,y) - 2Y(t,x,y) + Y(t,x-\Delta x,y)}{2\Delta x}\right) + \\ D\left(\frac{Y(t,x,y+\Delta y) - 2Y(t,x,y) + Y(t,x,y-\Delta y)}{2\Delta y}\right) + \mu Y(t,x,y) + \varepsilon_{xyt} \end{aligned} \tag{15.25}$$

This can be simplified by letting $\Delta t = 1$ and noting that $\Delta x = \Delta y$ so that we can express D in units of one cell width. This yields

$$\begin{aligned} Y(t+1,x,y) = Y(t,x,y) + \frac{D}{2}\left[Y(x+\Delta x,y,t) - 2Y(x,y,t) + Y(x-\Delta x,y,t)\right] + \\ \frac{D}{2}\left[Y(x,y+\Delta y,t) - 2Y(x,y,t) + Y(x,y-\Delta y,t)\right] + \mu Y(t,x,y) + \varepsilon_{xyt} \end{aligned} \tag{15.26}$$

Equation 15.26 cannot be solved without boundary conditions. If it represented a biophysical process, these would ordinary be a fixed set of values for either Y or its partial derivatives on the boundary. However, for our application we simply set the differentials δ_{xx} and δ_{yy} equal to zero outside the region on which Y is defined.

It is important to recognize that under ordinary circumstances, Equation 15.26 would be treated as an approximation to a parabolic partial differential equation and solved for $Y(x,y,t+1)$, the quantity of the left-hand side of Equation 15.26, by plugging. In our application, Equation 15.26 represents an approximation to the spatiotemporal autocorrelation structure and is solved for D and μ by plugging in the appropriate values of Y and minimizing the mean square error ε.

The set of normalized yield values in the code are represented by a 74 by 4 matrix called *Y.norm*. The code to carry out the solution is as follows.

```
> MSE <- function(P){
+   Yest <- numeric(74)
+   for (i in 1:74) Yest[i] <- Y1[i] +
+      P[1] * (dxx[i] + dyy[i]) + P[2] * Y1[i]
+   E <- sum((Yest[i] - Yp1[i])^2) / length(Y1)
+     return(E)
+}
>
```

```
> D <- numeric(3)
> mu <- numeric(3)
>
> for (n in 1:3){
+    Yp1 <- Y.norm[, n+1]
+    Y1 <- Y.norm[, n]
+    soln <- nlm(MSE, c(0,0))
+    D[n] <- soln[[2]][1]
+    mu[n] <- soln[[2]][2]
+ }
```

Here the function `nlm()` is a nonlinear minimization function (Press et al. 1986, p. 521) that computes the minimum value of the first argument using the second argument as the vector of initial values. Here are the results.

```
> D
[1] -0.3034430 -0.2006647 -0.3416246
> mu
[1] 0.4827022 0.3203651 0.7631263
```

Interestingly, the "diffusion coefficient" D takes on negative values. This is of course impossible in a physical system since it would contradict the second law of thermodynamics, but it is possible in this purely empirical curve fit. The values of D are fairly consistent for each of the three time periods. It is unclear whether this is anything more than a coincidence. Indeed, the results are sensitive to the method used to scale the Y values (Exercise 15.2).

Again, the results of this section do not add anything to our understanding of the processes underlying crop growth and development in Field 4.1. They serve strictly as an illustration of how an empirical model could be constructed for a spatiotemporal process. Cressie and Wrikle (2011, p. 332) encourage ecological modelers to build their models from first principles where possible. Nothing could be further from the truth regarding the model constructed in this section. The code written for the section, however, will work for a model that is developed from first principles with one very important exception: the boundary conditions. The model of Plant and Cunningham (1991) described in Section 15.3.1 used boundary conditions that specify zero flux as the spatial coordinates tend to infinity. The correct specification of boundary conditions is perhaps somewhat less crucial for a parabolic equation such as Equation 15.14 than it is for elliptic or hyperbolic partial differential equations, but it cannot be completely ignored. Those wishing to apply models developed from first principles to spatiotemporal processes in ecology are advised to consult a good text on the subject. My favorite is Street (1973).

15.4 Finite State and Time Models

15.4.1 Determining Finite State and Time Models Using Clustering

As an alternative to modeling the spatiotemporal structure of the data, one might consider a discrete time model in which each spatial location can move between finite states. The state and transition model (Westoby et al., 1989) is one such approach. It provides a framework for the modeling of successional dynamics in ecosystems. In the model, ecosystems can exist in

a set of discrete *states*, and *transitions* can occur between these states in finite time. Roel and Plant (2004a and 2004b) developed an exploratory method for organizing spatiotemporal data by organizing the data into clusters in space and time. In this section, we discuss a method used by for generating and analyzing such clusters. We apply this approach to the four years of normalized yield data in Field 4.1. The spatiotemporal method discussed in this section works by first forming clusters of the response variable in the data space of each year's normalized yield data. We demonstrate the method with the first two years of data, since these can be visualized, and then apply the method to the data from all four years. In Exercise 15.3, you are asked to analyze centered and scaled yield data, which provide a slightly different perspective.

The analysis uses the k-means clustering algorithm introduced in Section 12.6.1. It is important to note that no spatial information is used in the algorithm; the only values used are attribute values. This permits us to use the spatial data to assess the validity of the model. Clustering is carried out in R using the function `kmeans()`. Our two-year demonstration uses $k = 4$ clusters. First, we read in the data and set it up for the cluster analysis.

```
> Yield96 <- read.csv("created\\set4.1yld96ptsidw.csv", header = TRUE)
> Yield97 <- read.csv("created\\set4.1yld97ptsidw.csv", header = TRUE)
> Yield96.norm <- 100 * Yield96$Yield / max(Yield96$Yield)
> Yield97.norm <- 100 * Yield97$Yield / max(Yield97$Yield)
> Yield.2Yr <- data.frame(Yield96 = Yield96.norm,
+    Yield97 = Yield97.norm)
```

Generating the clusters just requires a simple call to `kmeans()`.

```
> set.seed(123)
> cl <- kmeans(Yield.2Yr, 4)
```

One way to visualize the clusters is in the data space, as shown in Figure 15.9. A second way is in geographic space, as shown in Figure 15.10. The relationship in the data space of Figure 15.9 appears roughly parabolic. In the geographic space of Figure 15.10, the clusters are spatially organized; that is, they appear to form a meaningful spatial pattern. This is not guaranteed to occur, and its occurrence may be taken as an indication that the clusters are biophysically meaningful. This will be discussed more fully below.

Cluster 1 contains points from the large northern area of the field that are low in both years. Cluster 2 is in the south and is the highest yielding region in 1996 but yields lower in 1997. Cluster 3 is located mostly in the center of the field and is low in 1996 and moderate in 1997. Cluster 4 is in the south-center and is the highest yielding region in 1997.

Now that we have an idea of the process, let's move on to the full four-year data set, in which we remove the southernmost two rows of sample points from the interpolated yield data. The data are loaded and organized into a data frame as in the previous sections. To begin, we must recognize that, since the clustering algorithm starts with a random set of seeds, the process will not always result in the same set of clusters. Let's see what happens when we generate nine replications of the clustering algorithm applied to the four-year yield data set. We will generate these replications of clusters with $k = 3$ and collect them in a cluster matrix called `CM` for display. Your results may be different from those displayed here due to slight differences in your interpolated yield values.

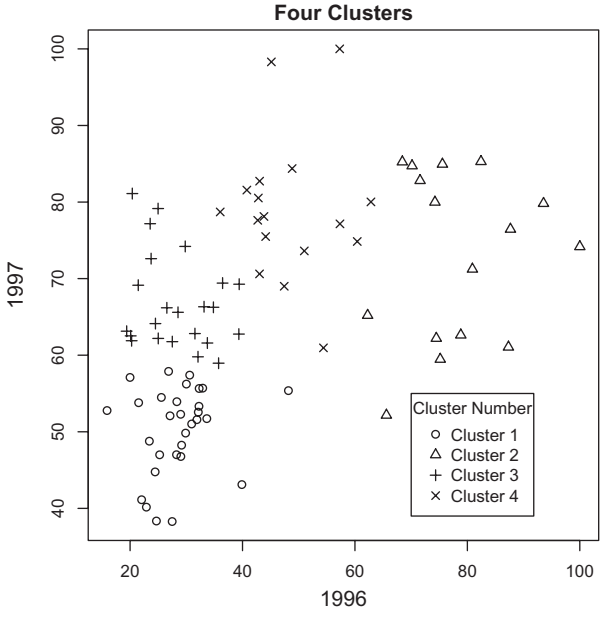

FIGURE 15.9
Plot of the data space of normalized yields in Field 4.1 in 1996 and 1997 showing the four clusters obtained via *k*-means clustering with $k = 4$.

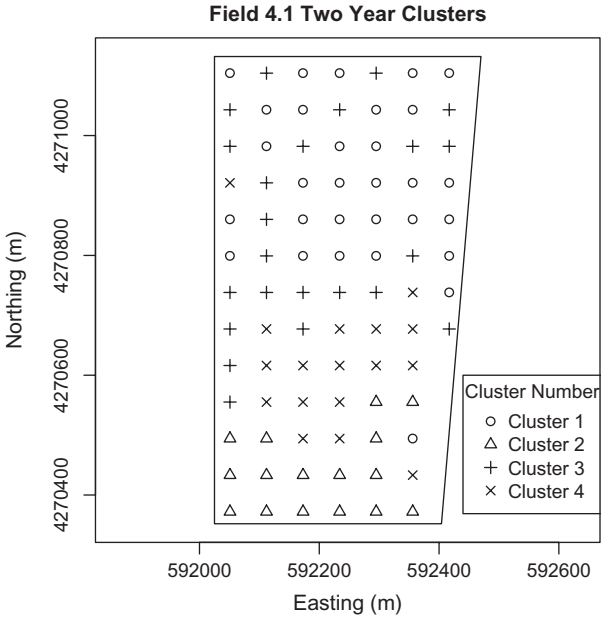

FIGURE 15.10
Thematic map showing the locations of the four clusters obtained via *k*-means clustering with $k = 4$.

```
> cluster.k <- 3
> n.reps <- 9
> n.sites <- nrow(Yield.4Yr)
> CM <- matrix(nrow = n.sites, ncol = n.reps)
> set.seed(123)
> for (i in 1:n.reps) CM[, i] <-
+     kmeans(Yield.4Yr, cluster.k)$cluster
> CM
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
 [1,]    1    3    1    3    3    3    1    2    3
 [2,]    1    3    1    3    3    3    1    2    3
 [3,]    1    3    1    3    3    3    1    2    3
 [4,]    1    3    1    3    3    3    1    2    3
 [5,]    1    3    1    3    3    3    1    2    3
 [6,]    1    3    1    3    3    3    1    2    3
 [7,]    1    3    1    3    3    3    1    2    3
 [8,]    3    1    2    1    2    2    3    1    1
   *    *    *  DELETED  *    *    *
[74,]    2    2    3    2    1    1    2    3    2
```

Since the initial cluster seeds are assigned randomly, the clusters are not identified by the same number in each replication. In addition, comparison of rows 7 and 8 of the matrix `CM` indicates that the *k*-means algorithm does not generate the same set of clusters in each replication. In dealing with spatial data, this property can be used to advantage. The objective in using cluster analysis with spatiotemporal data is to develop clusters that can be used to identify factors underlying the observed pattern in space and time of the response variable. That is, the clusters must be "biophysically meaningful." One criterion for being biophysically meaningful is that the cluster pattern should be *stable*, that is, it should be attainable starting from different sets of initial cluster seeds.

In order to test for stability it is necessary to align the cluster identification numbers. We will do this by minimizing the number of different ID values between them. For example, let's determine the permutation of cluster identification numbers in replication 2 (i.e., column 2 of the matrix `CM` above) that produces the best alignment with the identification numbers in each other replication. The first step is to generate a matrix called `perms` of all the possible permutations of the *k* identification numbers. This can be done using the function `permutations()` of the `e1071` package (Dimitriadou et al., 2017).

```
> library(e1071)
> print(perms <- t(permutations(cluster.k)))
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    2    2    1    3    3
[2,]    2    1    3    3    1    2
[3,]    3    3    1    2    2    1
```

Next, we create an array `test` to contain the permuted identification numbers. For example, suppose we want to permute the identification numbers in the second column of `CM` using the third permutation in the matrix `perm`.

```
> test <- numeric(nrow(CM))
> for (m in 1:nrow(perms)) test[which(CM[,2] == m)] <- perms[m,3]
> CM[,2]
> CM[,2]
```

```
 [1] 3 3 3 3 3 3 3 1 1 1 1 1 1 1 3 3 3 3 3 3 3 2 1 1 3 1 3 1 1 1 1 1
[33] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 1 2 2 2 1 3 2 2 2 2 2 1 2
[65] 2 2 2 2 2 2 2 2 2 2
> test
 [1] 1 1 1 1 1 1 1 2 2 2 2 2 2 2 1 1 1 1 1 1 1 3 2 2 1 2 1 2 2 2 2 2
[33] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 3 2 3 3 3 2 1 3 3 3 3 3 2 3
[65] 3 3 3 3 3 3 3 3 3 3
```

The 1's have been replaced by 2's, the 2's by 3's, and the 3's by 1's.

Suppose we want to determine the permutation of column 2 of the matrix `CM` that produces the best match with column 4. We will create a vector `diff.sum` to hold the number of elements of each permutation that differ from the corresponding element of column 4 of `CM`.

```
> diff.sum <- numeric(ncol(perms))
> for (n in 1:length(diff.sum)){
+   for (m in 1:nrow(perms)) test[which(CM[,2] == m)] <- perms[m, n]
+   diff.sum[n] <- sum((test != CM[,4]))
+}
> diff.sum
[1] 0 57 74 39 74 52
```

For my data set, Permutation 1 has the smallest sum of differences. Your results may vary.

We now embed this code in a function `min.sum()` that for each *i* and *j* returns the minimum sum of differences and the permutation that produces it.

```
> min.sum <- function(i, j, perms, CM){
+    test <- numeric(nrow(CM))
+    diff.sum <- numeric(ncol(perms))
+    for (n in 1:length(diff.sum)){
+ # test replaces the IDs in rep i with those in column n of perms
+    for (m in 1:nrow(perms)) test[which(CM[, j] == m)] <- perms[m, n]
+    diff.sum[n] <- sum((test != CM[, i]))
+    }
+ # which.min automatically selects the first in a tie
+    n.min <- which.min(diff.sum)
+    opt.id <- diff.sum[n.min]
+    return(c(n.min, opt.id))
+}
```

Now we are ready to compute the matrix of minimum ID differences and of optimal permutations for each pair of replications.

```
> dist.mat <- matrix(nrow = n.reps, ncol = n.reps)
> min.id <- matrix(nrow = n.reps, ncol = n.reps)
> for (i in 1:n.reps){
+    for (j in 1:n.reps){
+       x <- min.sum(i, j,perms, CM)
+       min.id[i, j] <- x[1]
+       dist.mat[i, j] <- x[2]
+    }
+}
> dist.mat
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
[1,]    0    0    2    0    0    0    0    0    2
[2,]    0    0    2    0    0    0    0    0    2
[3,]    2    2    0    2    2    2    2    2    0
[4,]    0    0    2    0    0    0    0    0    2
[5,]    0    0    2    0    0    0    0    0    2
[6,]    0    0    2    0    0    0    0    0    2
[7,]    0    0    2    0    0    0    0    0    2
[8,]    0    0    2    0    0    0    0    0    2
[9,]    2    2    0    2    2    2    2    2    0
> min.id
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
[1,]    1    6    4    6    3    3    1    5    6
[2,]    6    1    5    1    2    2    6    4    1
[3,]    4    3    1    3    6    6    4    2    3
[4,]    6    1    5    1    2    2    6    4    1
[5,]    5    2    6    2    1    1    5    3    2
[6,]    5    2    6    2    1    1    5    3    2
[7,]    1    6    4    6    3    3    1    5    6
[8,]    3    4    2    4    5    5    3    1    4
[9,]    6    1    5    1    2    2    6    4    1
```

Your matrix may be different from mine. The matrix of minimal distances is not always zero. Based on the matrix `dist.mat`, it appears that the replications divide into two groups. The alignment procedure is as follows. We first pick one replication to serve as the reference, and then we align all the other replications by minimizing the distance to this reference. We will pick replication 1.

```
> ref.rep <- 1
> best <- min.id[ref.rep,]
> CM.align <- matrix(nrow = n.sites, ncol = n.reps)
> for (i in 1:n.reps){
+    for (j in 1:cluster.k){
+      CM.align[which(CM[, i] == j),i] <- perms[j, best[i]]
+      }
+}
> CM.align
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
[1,]    1    1    1    1    1    1    1    1    1
[2,]    1    1    1    1    1    1    1    1    1
[3,]    1    1    1    1    1    1    1    1    1
  * * * DELETED * * *
[73,]   2    2    2    2    2    2    2    2    2
[74,]   2    2    2    2    2    2    2    2    2
```

Figure 15.11 shows the results. The clusters form into two primary groups. The first group in turn splits the northern, low-yielding area into two subgroups, corresponding to numbers 1 and 3 in the figure. The second group covers most of the southern section (number 2). The stability and spatial contiguity evident in the figure provides one indication that the clusters are biophysically meaningful.

A second check that the clusters are biophysically meaningful is that they must be spatially organized (i.e., arranged in a spatially meaningful pattern). To proceed further, and not have to rely on visualization, we will use a statistic that measures internal consistency

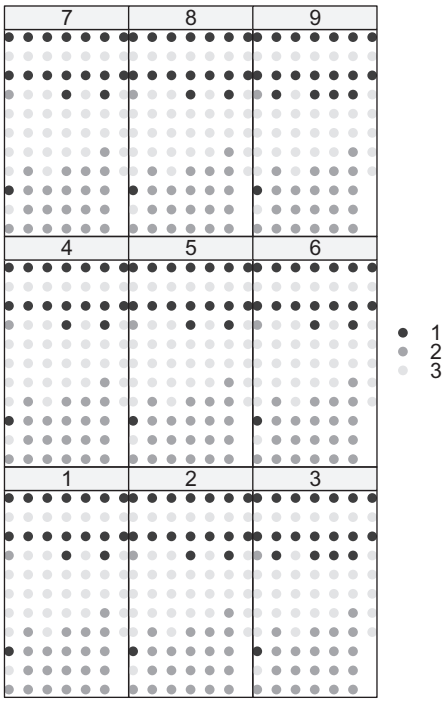

FIGURE 15.11
Thematic maps showing the locations of the clusters obtained via *k*-means clustering of four years of yield data with $k = 3$.

(Roel and Plant, 2004b). Suppose there are n data locations, and that the set under consideration has m replications (i.e., m of the random reorderings of initial seeds result in a member of this set). Let an indicator c_i, $i = 1,...,n$, be defined for each location i by

$$c_i = \begin{cases} 1 \text{ if in all members of the set location } i \text{ belongs to the same cluster} \\ 0 \text{ if at least one location } i \text{ belongs to a different cluster.} \end{cases}$$

Then the measure of consistency γ of the set is defined by

$$\gamma = \frac{\sum_{i=1}^{n} c_i - \frac{n}{m^k}}{n - \frac{n}{m^k}}. \tag{15.27}$$

The motivation for this definition of γ is similar to that for Cohen's κ (Cohen, 1960). The quantity n/m^k is the expected number of cells that all belong to the same cluster by chance. If all of the members of the set are identical then γ has the value 1, and if each cell of each member of the set is randomly assigned one of the k possible values, then $\gamma = 0$. The code to compute γ in Equation 15.27 is as follows:

```
> c.clus <- function(x) as.numeric(sum(abs(x[2:length(x)] - x[1]))
+     == 0)
> gamma.clus <- function(CM, set, k){
```

```
+    sum.c <- sum(apply(CM[, set], 1, c.clus))
+    m <- length(set)
+    n <- nrow(CM)
+    return((sum.c - n/m^k) / (n - n/m^k))
+}
```

With this as background, we are now ready to begin the analysis. The computations are identical to those just carried out, but now we set the number of clusters to $k = 2$ and the number of replications to $m = 25$.

```
> gamma.clus(CM.align, 1:ncol(CM.align), 2)
[1] 1
```

The γ statistic for $k = 2$ equals 1, indicating that all 25 replications are in alignment. Figure 15.12a indicates that the clusters are highly structured spatially. Figure 15.13a shows a plot of the mean normalized yield values. This indicates that average over the locations in cluster 2 in the southern part of the region is in every year higher than the regions in cluster 1 in the north.

Besides consistency from several initial cluster seeds and a spatial pattern, a third indication of whether the clusters are biophysically meaningful is whether they are hierarchically arranged. This question involves a bit of subtlety. Consider the case of an increase in the number of clusters from k to $k+1$. Suppose, for example, $k = 2$, so that we go from two to three clusters. Suppose that there is a single factor, say, soil clay content, underlying the separation of yield values into the two clusters. When the elements are partitioned into three clusters, the physical property underlying yield variability may continue to be the single factor clay content, it may switch to one or two completely new factors, or it may be comprised of clay content from the original two-cluster partition and some other factor, say, organic matter, within one of the two original clusters. It is only in the latter case that one would expect the clusters to be

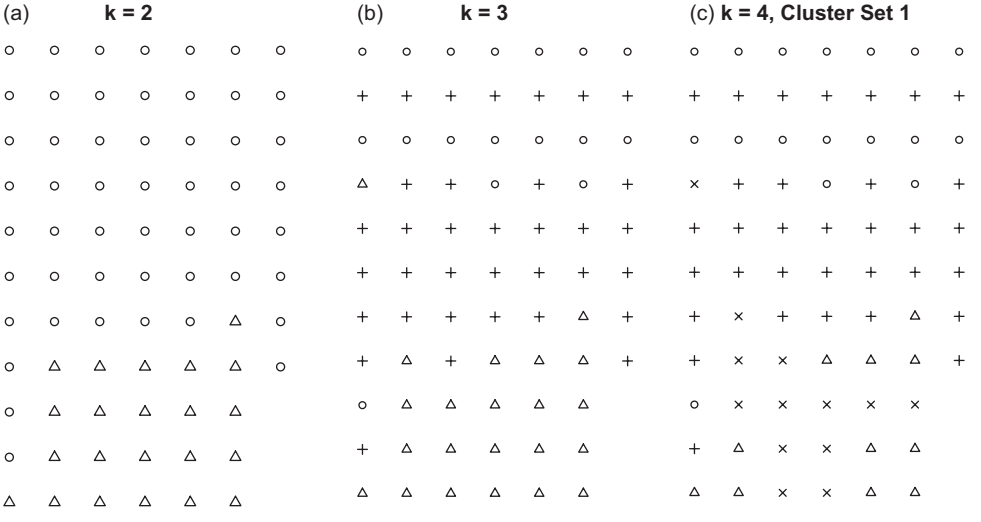

FIGURE 15.12
Locations of the clusters obtained using different values of k for the four years of yield data in Field 4.1. (a) $k = 2$; (b) $k = 3$; (c) $k = 4$.

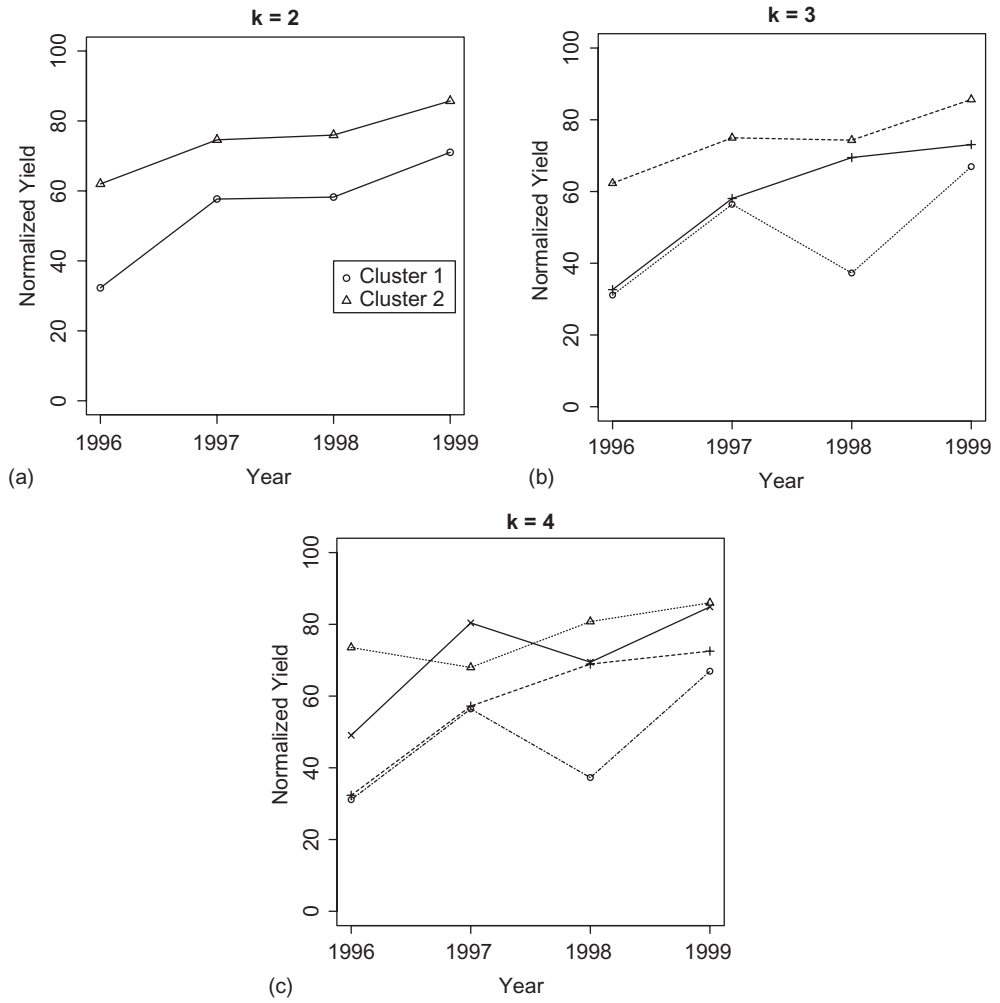

FIGURE 15.13
Plots of cluster mean yields over the four years for each of the cluster arrangements shown in Figure 15.5: (a) $k = 2$; (b) $k = 3$; (c) $k = 4$.

hierarchically arranged. Therefore, we will not test statistically for the presence or absence of a hierarchical arrangement, but rather simply examine the pattern of mean yields as k is increased. Let's increase k to 3 and see what happens.

```
> gamma.clus(CM.align, 1:ncol(CM.align), 3)
[1] 0.9729712
```

For $k = 3$ the γ statistic has the value 0.97, indicating that the replications again divide almost perfectly into two groups (your results may be different). The lower yielding northern cluster splits into two, one in the north and one in the center (Figures 15.12b and 15.13b). For $k = 4$ the γ value for the entire set is 0.66, indicating a loss of consistency. It turns out that the clusters can be divided into two groups that are internally consistent, and that these two groups are very similar. We only show the results for the first group. The higher yielding southern cluster splits into two (Figures 15.12c and 15.13c). The patterns of

cluster splitting do seem to indicate that the clusters are hierarchical. The number of data locations in each cluster for $k = 4$ has become sufficiently small that we will not carry out a cluster analysis for higher values of k. Therefore, we move on to the problem of determining the factors that underlie the observed cluster patterns.

15.4.2 Factors Underlying Finite State and Time Models

Our primary tools for attempting to determine the factors underlying the observed cluster patterns from Section 15.4.1 will be classification trees and random forest (Sections 9.3 and 9.4). Before carrying out this analysis, however, it is useful to examine climatic variables to determine what, if any, influence they may have had on these patterns. We have already seen evidence that the northern portion of the field suffered yield loss probably caused by aeration stress in the 1995–1996 season due to heavy rainfall. The file *dailyweatherdata.csv* was developed from data downloaded from the CIMIS (California Irrigation Management Information System) website, http://wwwcimis.water.ca.gov/cimis/welcome.jsp. The data were collected at that CIMIS weather station located on the UC Davis campus, which is located about 15 kilometers east of the fields in Data Set 4. Some values are missing from the data set.

```
> names(weather.data)
 [1] "Date"          "JulianDay"    "CIMIS_Eto_mm"  "Precip_mm"
 [5] "SolRad_W.sq.m" "AvgVap_kPa"   "MaxAirTemp_C"  "MinAirTemp_C"
 [9] "AvgAirTemp_C"  "MaxRelHum"    "MinRelHum"     "AvgRelHum"
[13] "DewPt_C"       "AvgwSpd_m.s" "WndRun_Km"     "AvgSoilTemp_C"
```

In order to compare the three summer seasons (remember that the wheat was grown in the winter), we will define the season of each year to begin on April 1 and end on September 1. This corresponds to the following.

```
> season.2 <- 548:701
> season.3 <- 913:1066
> season.4 <- 1278:1431
```

The quantity most likely to reflect the influence of climatic variability on the yield pattern is reference evapotranspiration `CIMIS _ Eto _ mm`.

```
> which(is.na(weather.data$CIMIS_Eto_mm))
integer(0)
> ETo1997 <- sum(weather.data$CIMIS_Eto_mm[season.2])
> ETo1998 <- sum(weather.data$CIMIS_Eto_mm[season.3])
> ETo1999 <- sum(weather.data$CIMIS_Eto_mm[season.4])
> c(ETo1997, ETo1998, ETo1999)
[1] 983.04 816.66 948.03
```

The third season, the summer of 1998, had a reference ET about 150 mm lower than the other two summer seasons. This indicates that the summer of 1998 may have been cooler than the other two. The usual way to measure seasonal temperature is with accumulated degree-days (Zalom et al., 1983). However, four days in the 1997 season and three in the 1998 season are missing daily minimum temperature data.

```
> which(is.na(weather.data[,8]))
[1] 353 648 649 650 651 652 653 1032 1349 1410
```

Before plotting the degree days, we will impute the missing values by linearly interpolating between the temperature values just before and just after each sequence of missing data. This interpolation is left for the Exercises (Exercise 15.5). After imputation, degree-days are computed using simple averaging with a lower threshold of 10°C.

```
> DD.per.day <- function(max.temp, min.temp) (max.temp+min.temp)/2 - 10
```

The function `cumsum()` is then used to generate the data to plot. Consistent with the evapotranspiration data, the highest degree-day accumulation occurs in 1997 (Figure 15.14a). Although the ultimate accumulation is about the same in 1998 as in 1999, the midseason is warmer in 1999. Precipitation is not really a factor in any season (Figure 15.14b, Exercise 15.6).

Now that we have developed the climatic data, we will go through successive analyses of the clusters generated with three successive values of k, beginning with $k = 2$. The code for the analysis is analogous to that used in Section 9.3. The output is heavily redacted to show only the first competitor and the first two surrogates at nodes where splits occur.

```
Node number 1: 74 observations, complexity param=0.5909091
  Primary splits:
      Clay < 35.52 to the right, improve=15.574660, (0 missing)
      SoilP < 7.21 to the left, improve=12.392600, (0 missing)
  Surrogate splits:
      Sand < 26.895 to the left, agree=0.932, adj=0.615, (0 split)
      SoilK < 156.45 to the right, agree=0.851, adj=0.154, (0 split)
Node number 2: 61 observations, complexity param=0.1818182
  Primary splits:
      SoilTN < 0.1065 to the left, improve=7.301125, (0 missing)
      SoilK < 230.9 to the left, improve=5.674194, (0 missing)
  Surrogate splits:
```

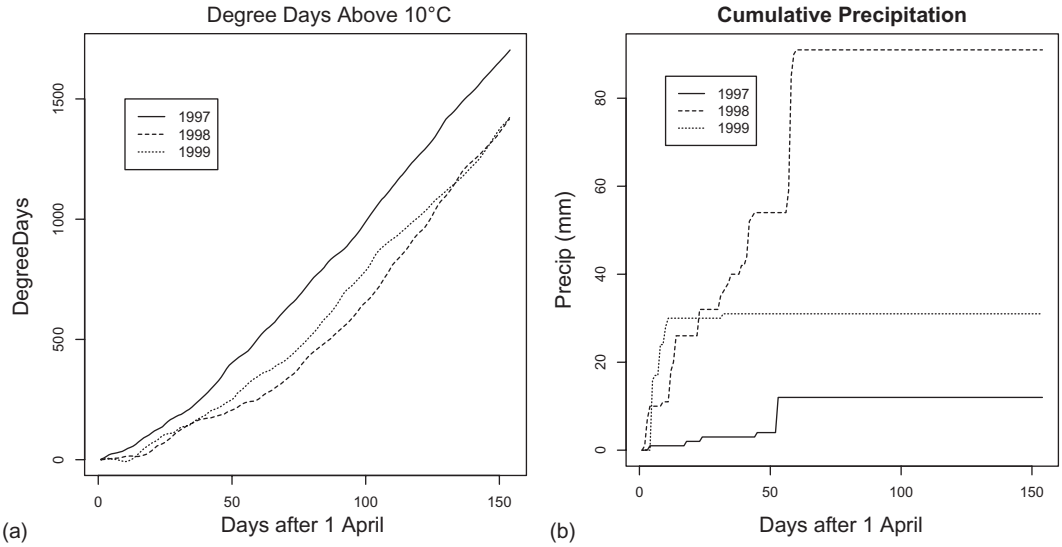

FIGURE 15.14
(a) Accumulated degree-days over 10°C in Fields 4.1 and 4.2 for the 1997, 1998, and 1999 seasons. (b) Cumulative precipitation in each season.

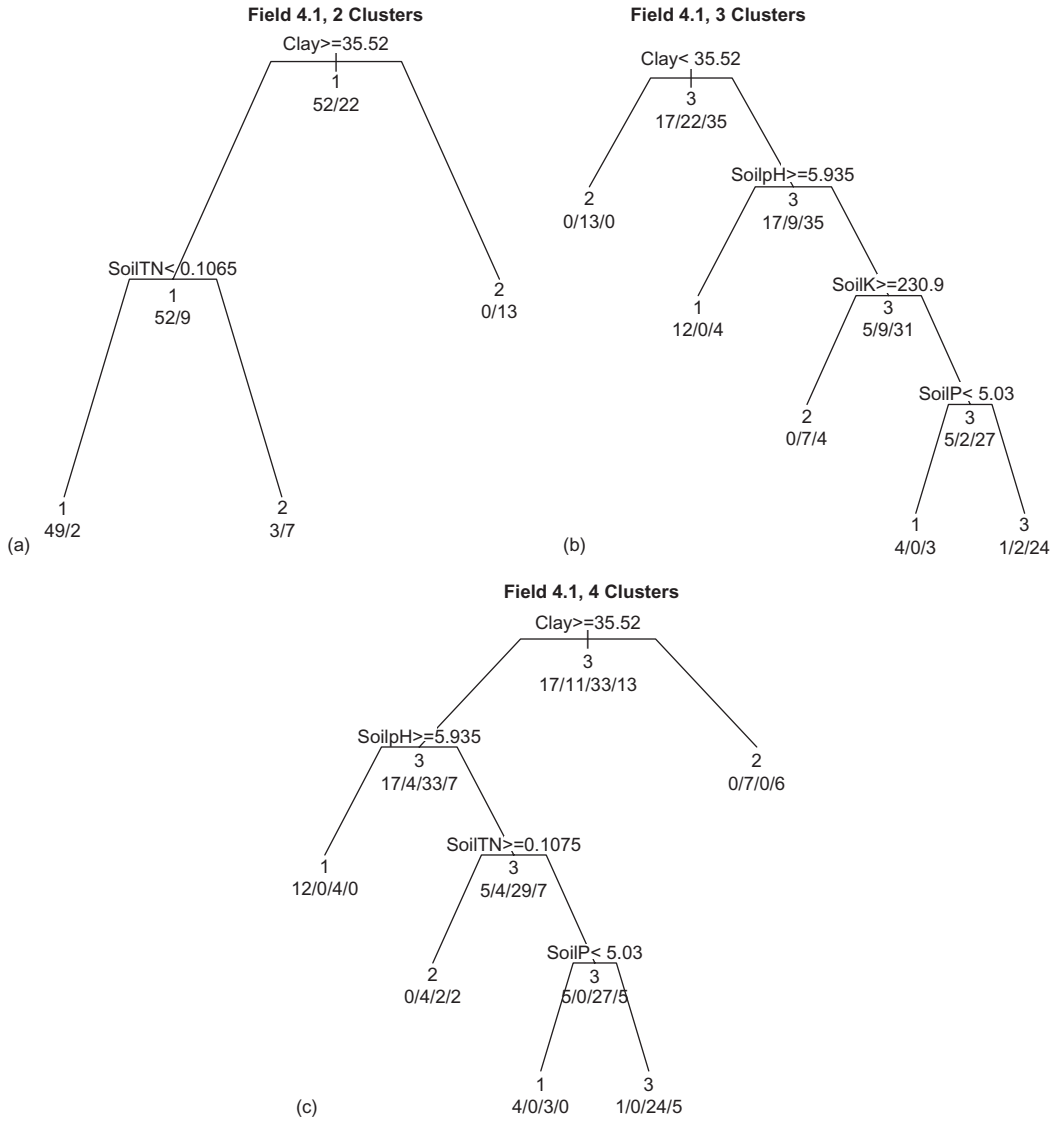

FIGURE 15.15
Classification trees with three different cluster values as the response variable for the clusters shown in Figure 15.12. (a) 2 clusters, (b) 3 clusters, (c) 4 clusters.

```
SoilP < 15.72 to the left, agree=0.869, adj=0.2, (0 split)
SoilK < 230.9 to the left, agree=0.869, adj=0.2, (0 split)
```

In the two-cluster set, cluster 1 is the northern, lower mean yield cluster and cluster 2 is the southern, higher yielding one (Figure 15.12a). The first splitting variable is *Clay*, with all high clay locations partitioned into a node identified as cluster 1 (Figure 15.15a). The second split, acting on the node defined by $Clay \geq 35.52$, is on *SoilTN*, with low values going into cluster 1. For $k = 3$ the higher yielding north splits on *SoilpH*, then on *SoilK*, and then on *SoilP* (Figure 15.15b). Based on the instability discussed in Section 9.3.5, we would

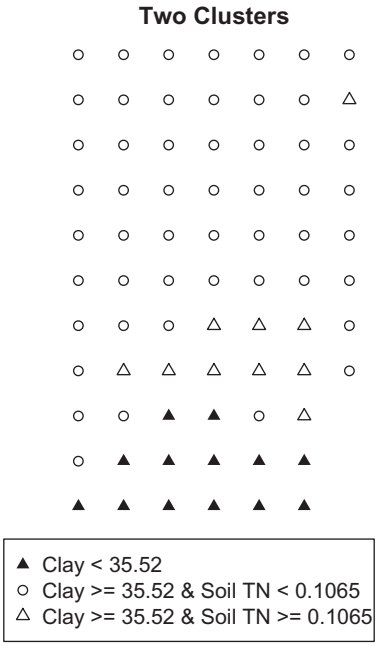

FIGURE 15.16
Plot of the nodes of the classification tree shown in Figure 15.15a. The white symbols are in the Cluster 1, the low mean yield cluster, and the black symbols are in Cluster 2.

not expect the lower splits to be very predictive. For $k = 4$ the same splits take place in the lower yielding region and no splits are identified in the south (Figure 15.15c). Figure 15.16 shows a map of the nodes of the $k = 2$ tree.

Figure 15.17 shows the results of a random forest analysis of the three-cluster data sets. Somewhat surprisingly, *Weeds* are rated fairly low. This may be because weeds only affect yield at high levels, primarily at *Weeds* = 5, and this only occurs in a relatively small number of locations. Thus, when these locations are not part of the random sample, weed level appears unimportant.

In Section 9.4.2, we saw that classification and regression trees may be unstable, that is, small changes in the variable values may have large effects on the tree. An alternative way of dealing with tree instability to those discussed in that section is to perturb the response variable slightly and test the effect of these perturbations on the tree structure. We will try this by raising or lowering the value of roughly 10% of the cluster numbers by one. We will also ensure that the cluster ID numbers are all between 0 and 4. We won't worry about the fact that some of the ID numbers will be both raised and lowered, with no net effect. Investigation of the `rpart` object using our old friend `str()` indicates that the first column of the data field `frame` contains the name of the splitting variable. As usual, see the R file for this section for the full code to set up the tree. Here is a quick simulation, showing only the first six nodes.

```
> n.clus <- 3
> data.Perturb <- data.Set4.1clus
> p <- 0.1
>set.seed(123)
>for (i in 1:10){
```

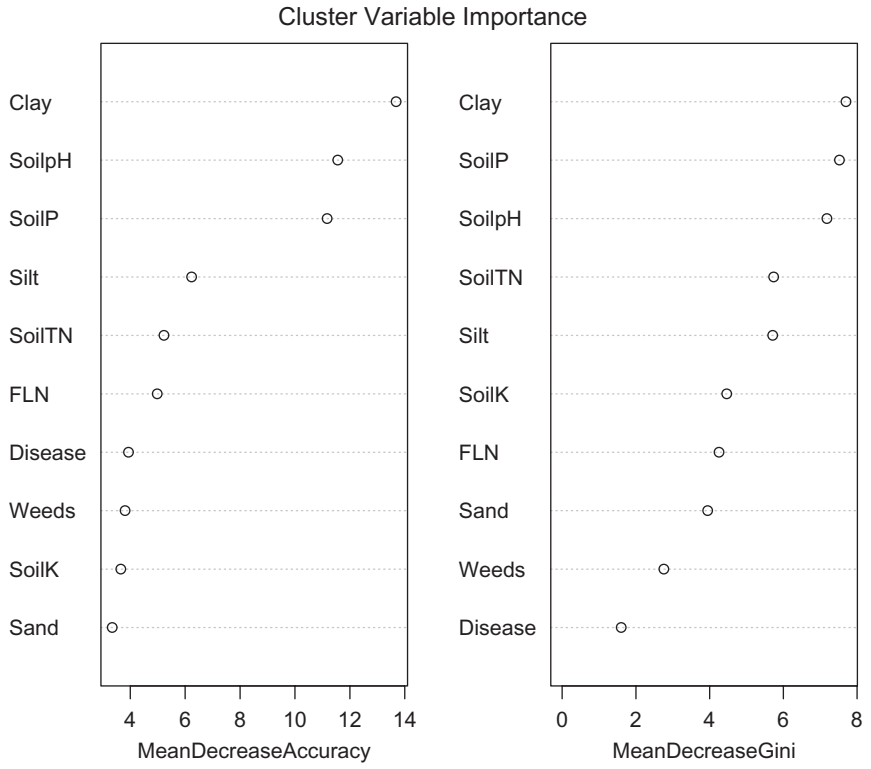

FIGURE 15.17
Variable importance plots from a random forest model of the $k = 2$ cluster data.

```
+    data.Perturb$Cluster <- clusters. Set4.1$x +
+       rbinom(74, 1, p) - rbinom(74, 1, p)
+    data.Perturb$Cluster[which(data.Perturb$Cluster < 0)] <- 0
+    data.Perturb$Cluster[which(data.Perturb$Cluster > 4)] <- 1
+    data.Perturb$Cluster
+    Perturb.rp <- rpart(model.1, data = data.Perturb,
+        method = "class")
+    print(as.character(Perturb.rp$frame[1:6,1]))
+}
[1] "Clay"    "<leaf>"  "SoilpH" "<leaf>"  "SoilK"  "Weeds"
[1] "Clay"    "<leaf>"  "SoilP"  "<leaf>"  "SoilpH" "Sand"
[1] "Clay"    "<leaf>"  "SoilP"  "<leaf>"  "SoilTN" "<leaf>"
[1] "SoilpH" "<leaf>"  "Sand"   "<leaf>"  "SoilP"  "<leaf>"
[1] "Clay"    "<leaf>"  "SoilTN" "<leaf>"  "Silt"   "SoilK"
[1] "Clay"    "<leaf>"  "SoilP"  "<leaf>"  "SoilpH" "<leaf>"
[1] "Clay"    "<leaf>"  "SoilpH" "SoilTN"  "<leaf>" "<leaf>"
[1] "Clay"    "<leaf>"  "SoilpH" "<leaf>"  "SoilTN" "<leaf>"
[1] "Clay"    "<leaf>"  "SoilTN" "SoilpH"  "<leaf>" "<leaf>"
[1] "Clay"    "<leaf>"  "SoilP"  "<leaf>"  "SoilpH" "<leaf>"
```

The printout shows the splitting variable at each node. The symbol `"<leaf>"` indicates that the node is a terminal node. *Clay* and soil nutrients are consistently identified as important.

Soil properties such as texture and organic matter content, and the content of nonlabile nutrients such as P and K, should not change much over years. In Exercise 15.4, you are asked to construct a scatterplot matrix of these quantities against yield of all four years. We will postpone an agronomic interpretation of these results until Chapter 17.

15.5 Bayesian Spatiotemporal Analysis

15.5.1 Introduction to Bayesian Updating

Bayesian statistical theory is based on the concept of subjective probability and on the combination of data with prior subjective probability to obtain posterior subjective probability. The concept of updating is central to the application of Bayesian theory to temporal data analysis. The Kalman filter (Gelb, 1974), which is used in optimal control of the trajectory of a dynamic system, and statistical decision theory (Raiffa and Schlaifer, 1961; Melsa and Cohn, 1978), which is concerned with the optimal use of observational data in the decision-making process, both make effective use of Bayesian updating.

To see how the updating process works, let's return to the example of Scientist A and Scientist B that we used in Section 14.1 to introduce subjective probability. Recall that Scientist A and Scientist B are both interested in determining the value of a parameter that we will call θ. Scientist A is about 95% certain that the value of θ lies between 860 and 940. This says that Scientist A's prior subjective probability distribution (assumed normal) has a mean $\hat{\theta} = 900$ and a standard deviation $\hat{\sigma} = 20$. A measurement of θ is made that results in a value plus or minus one standard deviation is $Y_1 = 850 \pm 40$. Equation 14.9 specifies the mean $\breve{\theta}$ and variance $\breve{\sigma}^2$ of the posterior $p(\theta \mid Y_1)$ as

$$\begin{aligned} \breve{\theta} &= \frac{(\hat{\tau}\hat{\theta} + \tau_1 Y_1)}{\hat{\tau} + \tau_1}, \quad \breve{\tau} = (\hat{\tau} + \tau_1) \\ \hat{\tau} &= \frac{1}{\hat{\sigma}^2}, \qquad \tau_1 = \frac{1}{\sigma_1^2}. \end{aligned} \tag{15.28}$$

For our data, Scientist A's updated parameters are $\breve{\theta} = 890$ and $\breve{\sigma} = 17.9$. The reduction in the standard deviation from $\hat{\sigma} = 20$ to $\breve{\sigma} = 17.9$ represents the increase in the certainty of Scientist A's subjective belief about the value of θ taking into account the data. We could now make a second observation Y_2 and repeat the process.

Before we go on, we must add a caveat. Equations 15.28 indicate that the only quantities affecting the change in standard deviation of Scientist A's subjective probability are the precision $\hat{\tau} = 1/\hat{\sigma}^2$ of the prior and the precision $\tau_1 = 1/\sigma_1^2$ of the data. While they have the nice, intuitively reasonable property that they predict an increased certainty of belief as more data are accumulated, they do not necessarily model the way a scientist would really think in all cases. For example, suppose that instead of the values given above, the observation is $Y_1 = 100 \pm 40$. According to Equations 15.28, Scientist A would dutifully modify his or her belief in the value of θ to $\breve{\theta} = 740$ with the same value for $\breve{\tau}$. In reality, however, Scientist A would probably consider the data value to be discordant and consider applying one of the options discussed in Section 6.2.1.

There are two fundamental issues at play here. The first is whether the Bayesian formulas represent an accurate model for how people modify their beliefs in the presence of new evidence, and the second is whether people, when confronted with new evidence, modify their beliefs in an appropriate way. These are both complex and controversial issues. Plant and Stone (1991, Ch. 3) provide an introductory discussion, and further references are provided in Section 15.6. Since this section is about the use of Bayesian inference, we will accept the assumptions of Bayesian updating of subjective probability and see where they lead.

We first consider a simple, schematic example of how Bayesian updating might be used to incorporate new data into a regression model. We begin with the same artificial data set introduced in Section 14.2 and shown in Figure 14.2. Since we are going to be basing a second regression on this regression, we will add the subscript 1 to the variables.

```
> library(R2WinBUGS)
> set.seed(123)
> n <- 20
> X1 <- rnorm(n)
> Y1 <- X1 + rnorm(n)
> print(coef(lm(Y1 ~ X1)), digits = 3)
(Intercept)          X1
    -0.0402      0.9217
```

Here are the results of the same Markov Chain Monte Carlo run as was done in Section 14.2.

```
> print(t(cbind(demo.sim1$mean, demo.sim1$sd)), digits = 3)
       beta0 beta1 tau   deviance
[1,] -0.0502 0.917 1.41  51.2
[2,] 0.199   0.206 0.467 2.62
```

The means and the inverse of the squares of the standard deviation of `beta0` and `beta1` can be used directly as the prior mean and precision in a second simulation, with one cautionary comment. Recall that the joint prior density $p(\beta, \tau)$ is the product of a normal prior for β and a gamma prior for τ. This is not a conjugate prior, however, so the posterior density $p(\beta, \tau \mid Y)$ does not have this form. Therefore, we do not actually use the posterior from one observation as the prior for the subsequent one. Instead, we formulate a new prior based on the statistical properties of the preceding posterior.

The random variable τ is distributed according to a gamma distribution with parameters μ and ν, where the mean is equal to μ / ν and the variance is equal to μ / ν^2. Let us denote the mean and standard deviation of τ by m_τ and sd_τ. Then we have $m_\tau = \mu / \nu$ and $sd_\tau^2 = \mu / \nu^2$. Therefore, $\nu = m_\tau / sd_\tau^2$ and $\mu = m_\tau \nu$. The values of these parameters are computed as follows:

```
> print(mu.0 <- demo.sim1$mean$beta0, digits = 3)
[1] -0.0502
> print(tau.0 <- 1/demo.sim1$sd$beta0^2, digits = 3)
[1] 25.3
> print(mu.1 <- demo.sim1$mean$beta1, digits = 3)
[1] 0.917
> print(tau.1 <- 1/demo.sim1$sd$beta1^2, digits = 3)
[1] 23.5
> print(nu.t <- demo.sim1$mean$tau / demo.sim1$sd$tau^2, digits = 3)
[1] 6.45
```

```
> print(mu.t <- demo.sim1$mean$tau * nu.t, digits = 4)
[1] 9.06
```

We will use these values in the prior distributions of a second data set, which in our example has a slightly different relationship between X and Y.

```
> set.seed(456)
> X2 <- rnorm(n)
> Y2 <- 0.7 * X2 + rnorm(n)
```

Here is the code that will be passed to WinBUGS. Only those parts that are changed from the first run are shown.

```
> demo.model <- function(){
+ beta0 ~ dnorm(mu.0, tau.0)
+ beta1 ~ dnorm(mu.1, tau.1)
+ tau ~ dgamma(mu.t, nu.t)
+ for (i in 1:n)
+   {
+   Y.hat[i] <- beta0 + beta1 * X[i]
+   Y[i] ~ dnorm(Y.hat[i], tau)
+   }
+}
> XY.data <- list(X = X2, Y = Y2, n = n,
+   mu.0 = mu.0, tau.0 = tau.0, mu.1 = mu.1, tau.1 = tau.1,
+   mu.t = mu.t, nu.t = nu.t)
```

Here are the results of the second simulation.

```
> print(t(cbind(demo.sim1.2$mean, demo.sim1.2$sd)), digits = 3)
     beta0  beta1 tau   deviance
[1,] -0.155 0.762 1.2   56.5
[2,] 0.144  0.134 0.29  1.78
```

The estimated slope decreases from 0.92 to 0.76 as a consequence of the new data, and the standard deviation of the posterior has declined from 0.21 to 0.13. In Exercise 15.9, you are asked to verify that this same result is obtained (approximately) if all of the data are used at once in a simulation with a noninformative prior.

Let us compare this result with that obtained by using ordinary least squares regression.

```
> print(coef(lm(Y1 ~ X1)), digits = 3)
(Intercept)          X1
    -0.0402      0.9217
> Y12 <- c(Y1,Y2)
> X12 <- c(X1,X2)
> print(coef(lm.12 <- lm(Y12 ~ X12)), digits = 3)
(Intercept)         X12
     -0.146       0.769
```

The results are almost identical. This is comforting, but you may be wondering what the point is of going to the trouble to use the Bayesian approach if we can get the same result using ordinary least squares (OLS). We will address this question by moving on to an example with real data.

15.5.2 Application of Bayesian Updating to Data Set 3

We have tentatively identified planting date *DPL*, irrigation effectiveness *Irrig* and nitrogen fertilizer rate *N* as the most important distinguishing factors in obtaining a high yield in east central Uruguay rice production (Data Set 3). Let us consider irrigation effectiveness. Improving irrigation effectiveness generally requires reworking the land to obtain a more uniform water depth given the topography of the field (see Figure 1.7). This requires a financial investment, and therefore a farmer will be interested in predicting the improvement in yield that would occur if the improvement was made. Probably not too many Uruguayan rice farmers (or any other farmers, for that matter) would use Bayesian updating to address this question, but let's suppose they did. In particular, consider the case of a farmer in the third year of the study who wants to use the results of the first two years to help predict the effect of an improvement in irrigation on the yield in his particular field. Our approach will follow that of Levy and Crawford (2009).

To study the procedure of Bayesian updating using Data Set 3, we will break up the data by season. As usual we will work with normalized yield.

```
> data.Set3$YieldN <- data.Set3$Yield / max(data.Set3$Yield)
> data.Set3S12 <- data.Set3[-which(data.Set3$Season == 3),]
> data.Set3S1 <- data.Set3[which(data.Set3$Season == 1),]
> data.Set3S2 <- data.Set3[which(data.Set3$Season == 2),]
> data.Set3S3 <- data.Set3[which(data.Set3$Season == 3),]
> data.Set3F3S2 <- data.Set3S2[which(data.Set3S2$Field == 3),]
> data.Set3F3S3 <- data.Set3S3[which(data.Set3S3$Field == 3),]
```

We will consider the specific case of Field 3. This field was farmed in both the second and third seasons by Farmer *B*. The irrigation effectiveness in the field was not uniformly good (Exercise 15.10), and the mean yield was lower in the regions whose irrigation effectiveness was lower.

```
> with(data.Set3F3S2, tapply(Yield, Irrig, mean))
       3        4
8046.667 8712.562
```

Will investing in improved irrigation increase yield? We will use Bayesian analysis to incorporate the experience of the other farmers to address this question. Our treatment will of course be very superficial. Readers who are interested in addressing this issue in greater depth are referred to the sources in Section 15.6.

The key idea is to employ Bayes' theorem to quantify the experiences of Farmer *B* as well as the other farmers and to use this experience to formulate a subjective probability distribution for the effect of irrigation effectiveness on rice yield. This requires that we treat the ordinal scale variable *Irrig* as if it were measured on a ratio scale, and that we assume that *YieldN* is linearly related to *Irrig*, at least in the range of values of *Irrig* that characterizes our data set. Considering the difficulties encountered in Section 14.5.2, and the results of Section 12.3, we will not incorporate spatial autocorrelation into the model.

Let's start by examining the relationship between *YieldN* and *Irrig* in Season 2 using OLS regression.

```
> model.lm <- lm(YieldN ~ Irrig, data = data.Set3F3S2)
> summary(model.lm)
Coefficients:
```

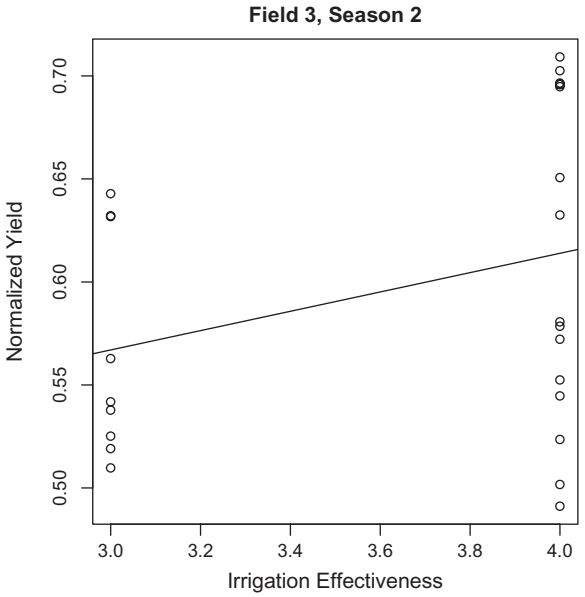

FIGURE 15.18
Plot of *Yield* vs. *Irrig* for Field 3 in Season 2 of the data of Data Set 3, showing the OLS regression line.

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.42622    0.10908    3.908 0.000708 ***
Irrig       0.04692    0.02971    1.579 0.127919
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.0713 on 23 degrees of freedom
Multiple R-squared: 0.09784,     Adjusted R-squared: 0.05861
F-statistic: 2.494 on 1 and 23 DF, p-value: 0.1279
```

The regression coefficient b_1 is not significantly different from zero. The low R^2 indicates a high variability in the data, which is confirmed in Figure 15.18. The figure also indicates, however, that there are two influence points at *Irrig* = 3.

Now we will run a Bayesian analysis of the data using a noninformative prior. First, we set up the noninformative prior model.

```
> Set3model.ni <- function(){
+ beta0 ~ dnorm(0, 0.001)
+ beta1 ~ dnorm(0, 0.001)
+ tau ~ dgamma(0.01, 0.01)
+ for (i in 1:n)
+   {
+   Y.hat[i] <- beta0 + beta1*Irrig[i]
+   Yield[i] ~ dnorm(Y.hat[i], tau)
+   }
+}
```

Next, we set the data and inits.

```
> XY.data.F3S2 <- list(Irrig = data.Set3F3S2$Irrig,
+   Yield = data.Set3F3S2$YieldN, n = nrow(data.Set3F3S2))
> XY.inits <- function(){
+    list(beta0 = rnorm(1), beta1 = rnorm(1), tau = exp(rnorm(1)))}
```

Now we run the model. The test and coda steps are not shown.

```
> Set3.sim. F3S2ni <- bugs(data = XY.data.F3S2, inits = XY.inits,
+   model.file = paste(mybugsdir,"\\Set3model.ni.bug", sep = ""),
+   parameters = c("beta0", "beta1", "tau"), n.chains = 5,
+   n.iter = 10000, n.burnin = 1000, n.thin = 10,
+   bugs.directory = mybugsdir)
> print(Set3.sim. F3S2ni, digits = 2)
Inference for Bugs model at "c:\aardata\book\Winbugs\Set3model.ni.bug",
fit using WinBUGS,
5 chains, each with 10000 iterations (first 1000 discarded), n.thin = 10
n.sims = 4500 iterations saved
           mean   sd   2.5%      25%    50%     75%  97.5%   Rhat n.eff
beta0      0.43  0.12 0.18      0.36   0.44    0.52   0.67   1.01   310
beta1      0.04  0.03 -0.02     0.02   0.04    0.07   0.11   1.01   300
tau      169.13 50.01 86.95  133.10 164.25 198.90 279.20   1.00  2600
deviance -59.46  2.93 -62.88 -61.61 -60.20 -58.09 -51.95   1.00  1200
```

The values of the regression coefficients are roughly the same as those of the OLS regression. The credibility interval includes zero, so Farmer B's subjective probability could not exclude a lack of improvement of yield following an increase in irrigation effectiveness.

Now suppose that Farmer B wants to incorporate the experience of other farmers into his subjective probability prior to making a decision at the start of Season 2. First, we must compute the parameters of the posterior from Season 1 to use in the prior for Season 2.

```
> XY.data.S1 <- list(Irrig = data.Set3S1$Irrig,
+ Yield = data.Set3S1$YieldN, n = nrow(data.Set3S1))
```

Now we run the model with a noninformative prior.

```
> Set3.sim. S1ni <- bugs(data = XY.data.S1, inits = XY.inits,
+ model.file = "c:\\aardata\\book\\Winbugs\\Set3model.ni.bug",
+ parameters = c("beta0", "beta1", "tau"), n.chains = 5,
+ n.iter = 10000, n.burnin = 1000, n.thin = 10,
+ bugs.directory = mybugsdir)
> print(Set3.sim. S1ni, digits = 2)
Inference for Bugs model at "c:\aardata\book\Winbugs\Set3model.ni.bug",
it using WinBUGS,
5 chains, each with 10000 iterations (first 1000 discarded), n.thin = 10
n.sims = 4500 iterations saved
            mean    sd    2.5%      25%     50%     75%    97.5% Rhat n.eff
beta0       0.22  0.03    0.16     0.20    0.22    0.24     0.29 1.01   700
beta1       0.06  0.01    0.04     0.05    0.06    0.06     0.08 1.01   590
tau       192.72 24.65  146.89  175.70  191.60  208.92   242.45 1.00 2300
deviance -305.71  2.67 -308.70 -307.70 -306.40 -304.50 -298.90 1.01   930
```

The mean value of $\hat{\beta}_1$ over all of the fields in the first year is 0.06 and the credibility interval, which is much smaller because there are more data, does not include 0.

Next, we compute the parameter estimates of the posterior distributions for β_0, β_1, and τ.

```
> print(mu.0 <- Set3.sim. S1ni$mean$beta0, digits = 3)
[1] 0.222
> print(tau.0 <- 1/Set3.sim. S1ni$sd$beta0^2, digits = 3)
[1] 973
> print(mu.1 <- Set3.sim. S1ni$mean$beta1, digits = 3)
[1] 0.0585
> print(tau.1 <- 1/Set3.sim. S1ni$sd$beta1^2, digits = 3)
[1] 11818
> print(nu.t <- Set3.sim. S1ni$mean$tau / Set3.sim. S1ni$sd$tau^2,
+   digits = 3)
[1] 0.316
> print(mu.t <- Set3.sim. S1ni$mean$tau * nu.t, digits = 3)
[1] 60.9
```

Now we will set up a model that incorporates the parameters of this posterior into the prior for Season 3.

```
> Set3.model.ip <- function(){
+ beta0 ~ dnorm(mu.0, tau.0)
+ beta1 ~ dnorm(mu.1, tau.1)
+ tau ~ dgamma(mu.t, nu.t)
+ for (i in 1:n)
+   {
+   Y.hat[i] <- beta0 + beta1 * Irrig[i]
+   Yield[i] ~ dnorm(Y.hat[i], tau)
+   }
+}
```

Next, we set the data to include the prior parameters.

```
XY.data.F3S2ip <- list(Irrig = data.Set3F3S2$Irrig,
  Yield = data.Set3F3S2$YieldN, n = nrow(data.Set3F3S2),
  mu.0 = mu.0, tau.0 = tau.0, mu.1 = mu.1, tau.1 = tau.1,
  mu.t = mu.t, nu.t = nu.t)
```

Now we are ready to run the model.

```
> Set3.sim. F3S2ip <- bugs(data = XY.data.F3S2ip, inits = XY.inits,
+     model.file = "c:\\aardata\\book\\Winbugs\\Set3model.ip.bug",
+     parameters = c("beta0", "beta1", "tau"), n.chains = 5,
+     n.iter = 10000, n.burnin = 1000, n.thin = 10,
+     bugs.directory = mybugsdir)
> print(Set3.sim. F3S2ip, digits = 2)
Inference for Bugs model at "c:\aardata\book\Winbugs\Set3model.ip.bug",
it using WinBUGS,
5 chains, each with 10000 iterations (first 1000 discarded), n.thin = 10
n.sims = 4500 iterations saved
           mean    sd    2.5%    25%     50%    75%  97.5%   Rhat  n.eff
beta0      0.30  0.02    0.25   0.28    0.30   0.32   0.35      1   3400
beta1      0.08  0.01    0.06   0.07    0.08   0.08   0.09      1   2200
tau      191.98 22.73 149.75 176.30 191.40 206.72 239.00      1   4500
deviance -59.84  2.26 -62.25 -61.34 -60.51 -59.05 -53.52      1   4500
```

The estimated mean goes from $\hat{\beta}_1 = 0.04$ to $\hat{\beta}_1 = 0.08$, and the credibility interval no longer includes zero.

Suppose Farmer B does not make any irrigation improvements in Season 2 but decides to explore the possibility again in Season 3 (i.e., the subsequent season in which the field is planted to rice). He has to decide which farmers to include in his data set. One possibility would be to only incorporate data from Season 2 from his own field. This, however, would throw away information from the other farms, and one of the advantages of the Bayesian approach is precisely this capacity to incorporate such information (Hilborn and Mangel, 1997, p. 203). Farmer B could incorporate only the data from other northern farmers, under the assumption that they are more like him than the others (Exercise 15.11). Alternatively, he could incorporate data from all the farms. Another alternative would be to somehow weight the data, or simply to make up his own parameter values. Since he is describing his own subjective probability, there are no rules that he must follow, and that is a difficulty. When one assigns subjective probabilities, if the assignments are made in an arbitrary manner, then it is easy to slip, possibly subconsciously, into a practice of assigning these probabilities in a way that confirms the desired outcome, that is, of data snooping (Walters, 1986, p. 166). If this methodology is to be applied sensibly, rules must be established in advance and strictly followed.

15.6 Further Reading

From the practitioner's point of view, the literature on spatiotemporal problems is still fairly sparse. Cressie and Wikle (2011) is very comprehensive, but the mathematical level may be a challenge for some practitioners. Hengl et al. (2008) provide case studies of spatiotemporal modeling. Fortin and Dale (2005) devote a chapter to the topic, including some material on spatiotemporal clustering and animal movement modeling. LeSage and Pace (2009) also include a chapter on spatiotemporal models. In addition to `spacetime`, the R package `Stem` (Cameletti, 2011) deals with spatiotemporal modeling, including kriging and bootstrapping.

Okubo (1980) is the standard source for dispersal. See for example, Dennis et al. (1995) or Cantrell and Cosner (2003) for a discussion of diffusion-reaction models in ecology. For another example of a state and transition model, see Whalley (1994). George et al. (1992) and Huntsinger and Bartolome (1992) developed state and transition models for oak woodlands of the type described by Data Set 1. These models were studied in a spatial context by Plant et al. (1999). As an alternative, Alonso (2008) developed a state and transition simulation system using Bayesian networks (Jensen, 2001). Wiles and Brodahl (2004) and Ferraro et al. (2012) apply recursive partitioning to clusters in a similar way to that done in Section 15.4.2.

Although it is now a bit dated, Walters (1987) is still an excellent source for Bayesian updating. Hilborn and Mangel (1997) also has some very good information. Duncan et al. (1995) provide an example of Bayesian updating in a social science setting. Pearl (1986) provides a good discussion of Bayesian inference. The Dempster-Schafer theory of evidence (Dempster, 1990) is a major competitor of Bayesian updating. See also Cohen (1985). Banerjee et al. (2004) devote a chapter to Bayesian analysis of spatiotemporal data. They focus on the use of a matrix transformation called the singular value decomposition to reduce the number of variables in the analysis.

Exercises

15.1 The analyses in this chapter used adjusted values of *Yield* because of the large variation in mean and variance between years. Many of the results of this chapter are sensitive to how the data are adjusted to account for this variation. In the text the data were normalized, and in the exercises the same data will be analyzed using data that have been centered and scaled. Repeat the construction of the spatiotemporal semivariograms carried out in Section 15.2.2. with data that have been centered and scaled using the function `scale()`.

15.2 Repeat the process analysis of Section 15.3.1 using centered and scaled values. Note that the resulting estimates of the diffusion coefficient D are no longer aligned.

15.3 Re-run the cluster analysis of two clusters using centered and scaled values of *Yield*. (a) Create a plot of the cluster means similar to Figure 15.12a. What new information does this provide? and (b) Create a plot showing the scaled yield values at each location, with `pch` values chosen so you can distinguish the clusters.

15.4 Soil texture, organic carbon, and the mineral nutrients K and P should remain fairly stable over a four-year period in Field 4.1. Create a scatterplot matrix of *Yield* in each year against these quantities.

15.5 Some of the weather data from the CIMIS file http://wwwcimis.water.ca.gov/cimis/welcome.jsp are missing. Use linear regression of nearby data to impute these values.

15.6 Create the plot of cumulative precipitation in the fields in Data Set 4 shown in Figure 15.14b.

15.7 Plot individual regression trees for *Yield* in Field 4.1 in each year against the same explanatory variables as you used in Exercise 15.2. Do you see any patterns emerging?

15.8 Create a plot of the means by year of normalized yield over each of the terminal nodes of the classification tree of the two-cluster model (Figure 15.12a). What can you infer from the plot?

15.9 In Section 15.5.1, a Bayesian regression model was computed for a set of artificial data, and then the model was updated using a second set of artificial data. Using the same data, verify that approximately the same results are obtained if all of the data is used at once with a noninformative prior.

15.10 Create a map showing the irrigation effectiveness of Field 3 in Season 2. What does this say about the economics of Farmer B's decision as to whether to improve the irrigation effectiveness in this field?

15.11 Suppose Farmer B does not make any irrigation improvements in Field 3 in Season 2. (a) Estimate the regression coefficient for *Irrig* based on a noninformative prior. (b) Estimate the regression coefficient for *Irrig* based on an informative prior using only data from Field 3 in Season 2. (c) Estimate the regression coefficient for *Irrig* based on an informative prior using data from all of the northern fields in Season 2.