6

Preparing Spatial Data for Analysis

6.1 Introduction

Georeferenced data usually require considerable manipulation and checking before they can be subjected to statistical analysis. Point sample data, whether they are manually sampled (e.g., soil core data) or automatically sampled (e.g., yield monitor data) must often be converted from a spreadsheet or database format into a geographic information system (GIS) data file format such as the ESRI shapefile. Automatically sampled data often contain numerous outliers that must be detected and dealt with. Image data must often be georegistered to the earth's surface. Moreover, spatial data are often *misaligned*, that is, they are not recorded at the same location and, in a sense that will be made clearer later in this chapter, they are often measured at different scales.

One of the issues that must be decided in the initial data processing phase is the projection in which to represent the data. In the northern hemisphere, the two most common choices are longitude-latitude and Universal Transverse Mercator (UTM) coordinates. This book does not provide any discussion of geographic coordinate systems. A complete discussion of this topic is given by Lo and Yeung (2007, Chapter 2). Most global positioning systems permit the user to select either longitude-latitude in the WGS 84 datum (Lo and Yeung, 2007, p. 40) or UTM. The advantage of working in UTM is that it is a conformal projection (Lo and Yeung, 2007, p. 43), that is, shapes of reasonably small areas on the ground are preserved on the map. A major disadvantage of UTM coordinates is that the UTM zones are only six degrees of longitude in width, and as a result, many geographic features span more than one zone. For example, Data Set 2 is located in UTM Zones 10 and 11. As a general rule, it is good practice when recording data in the field to record in the same projection as that of other layers in the data set, or, if these are in different native projections, to record in the same projection as that data set considered the most geographically accurate. R provides the capability to transform the projection of a data set (Section 2.4.3). Also, the functions in the `sf` and `sp` packages can correct for the spherical shape of the earth when performing distance calculations using longitude and latitude.

As was mentioned in Chapter 1, the intent of the book is to mimic the process of analysis of a real data set, using the four examples that we carry along. After four introductory chapters and a chapter on data collection, we now begin this simulated data analysis project. The objective in this chapter is to first evaluate the data sets for quality and second to put them into a form suitable for further analysis. The spatial data are assumed to be error free, so the quality evaluation focuses on the attribute data. The process of putting the data into a form suitable for further analysis, however, generally focuses on the spatial component.

Data are often represented as a table in which the columns are the *data fields* and the rows are the *data records* (Section 2.4.1, Lo and Yeung, 2007, p. 76). In this representation, a data field is a specific attribute item such as mean annual precipitation, and a data record is the collection of the values of all the data fields for one occurrence of the data. We begin in Section 6.2 with a discussion of quality control of attribute data. Sections 6.3 and 6.4 deal with the spatial component. Section 6.3 presents a brief discussion of geostatistical interpolation procedures to estimate its attribute values at different locations. Section 6.4 deals with the application of these procedures to the resolution of misalignment problems.

6.2 Quality of Attribute Data

6.2.1 Dealing with Outliers and Contaminants

"A weed is a…plant out of place" (Klingman, 1961, p. 1). Data quality control is like weed management: one must identify and deal with data values that are out of place. However, unlike weeds, it is often difficult to tell whether a data value is out of place or not, or even what it means to be out of place. In thinking about data quality, we must first consider the sources of data variability. Anscombe (1960, see also Barnett and Lewis, 1994, p. 33) identifies three sources of such variability. For our purposes, we will paraphrase Anscombe's characterization slightly. The sources are (1) *inherent variability*, the variability that would be observed in the population even if all measurements were perfectly accurate; (2) *measurement error*, error in the reading of the measurement instrument; and (3) *execution error*, which we define as the inclusion of a valid record from a population other than the one we intend to measure. There are a number of well-known cases, such as the discovery of penicillin and the discovery of the ozone hole, in which data values were initially rejected as belonging to category (2), measurement error, when indeed they belonged to category (3), observation of a different, previously unknown population, and the previously unknown population that they represented turned out to be extremely important. Occurrences such as this are often cited as reasons that data should never be completely discarded without a trace. Nevertheless, some data values warrant special attention.

In thinking about data quality, it is important to recognize the difference between *outliers* and *contaminants*. Barnett and Lewis (1994, p. 7) define an outlier to be "an observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data." They define a contaminant to be a data value coming from a distribution other than the one characterizing the population being measured (i.e., a member of category (3) of the previous paragraph). An outlier might not be a contaminant (it may be an error, or it may be a valid member of the population with a one-in-a-million value), and a contaminant might not be an outlier (it may come from a different population but have a value consistent with members of the population we intended to measure).

Data quality control involves two phases: identifying data values that warrant special attention, either as potential outliers or as potential contaminants or both; and determining what action to take in response to these identified special values. These two phases should not be considered separately, because the decision of which data values to identify as special depends partly on the intended fate of these values. Values that are identified as warranting special attention will be called *discordant* values (Barnett and Lewis, 1994).

In deciding what to do with a data value that has been identified as discordant, one may consider four options: (1) include it in the analysis, possibly mentioning it in the discussion and characterizing its effect; (2) remove it from the analysis (possibly to become part of a different analysis); (3) modify its value; and (4) modify the analysis or analytical methods because of its presence. Not all special values will necessarily receive the same treatment, and indeed the treatment a data value receives depends partly on the determination of how the variable is characterized, and partly on the objectives of the data analysis. Kruskal (1960) gives a vivid if militaristic example of the importance of considering the objectives. Suppose 50 bombs are dropped on a target, and that a few go wildly astray. Suppose further that the bombs that go astray are observed to do so because their fins came loose. If we are concerned with the accuracy of the whole bombing system, then these wild bombs should be included in the analysis. If, however, the concern is with the accuracy of the bombsight, then the wild bombs should be excluded.

Most of the analyses described in this book have as their ultimate objective the determination of the relationship between observed environmental factors and some response variable such as species presence or crop yield. It is well known that certain statistics, such as the sample mean, are highly sensitive to extreme values in the data. Therefore, even if an extreme value may be the result of inherent variability in the data (the first and seemingly most legitimate of the three sources of data variability given above), including that value in the analysis may result in an unacceptably large error in the estimate of a population parameter. This error might provide justification for eliminating such a data record from the analysis (option (2) above). Removing a data record because of its effect is, however, a form of data censorship (or data snooping). It may be preferable to simply remove a certain number or fraction of the most extreme values prior to the analysis without regard to their actual magnitude. Statistics such as the *trimmed mean* are manifestations of this concept of automatic removal of the most extreme values. The trimmed mean is computed by automatically removing ("trimming") a pre-specified number or fraction of the highest and lowest data values from the computation of the mean.

It sometimes may happen that we do not want to discard an extreme value completely. This may be because doing so would change the number of observations, and we do not want to do this, or it may be because, although the value of the observation is considered too extreme, the observation's sign and the fact that the observation has a large magnitude are considered valid. A common practice in this case is to retain the data record but modify its value (option (3) above). A widely used data modification method is *Winsorization* (Barnett and Lewis, 1994, p. 78), in which the value of an observation is replaced by its nearest neighbor in an ordered list of the data. For example, in a data set consisting of the five values 3, 6, 4, 25, and 2, the data record whose value is 25 would be retained, but its value would be replaced with 6. Other similar methods are given by Barnett and Lewis (1994).

If the decision on how to proceed with the data is to adopt option (4) above and accommodate the discordant value or values, then one must determine what their effect on the analysis might be and how it can be ameliorated. A common strategy is the use of resistant statistical methods (Barnett and Lewis, 1994, Ch 3), that is, methods that are not sensitive to extreme values. For example, if we are interested in determining a measure of central tendency, then the fact that the mean is sensitive to extreme values suggests that we use the median instead.

Even if the data do not include any discordant values, they may have a different problem. Many of the spatial analysis methods described in this book assume that the data consist of random variables with a normal probability distribution and are sensitive to this assumption to some degree. If the data do not appear to be normally distributed, one may decide to transform them. Alternatively, one may wish to employ robust statistical methods, that is, methods that are not sensitive to the distributional assumption. The adoption of option (4), for example, by electing to use robust or resistant methods, represents a response to data quality problems by accommodating them and will not be discussed extensively in this book (the nonparametric methods described in Chapter 9 are, however, generally more robust than the parametric methods described in the rest of the book). The remainder of the present section will deal with options (2) and (3), in which the data values must be removed or modified in some way to make them conform to the assumptions of the statistical analysis.

6.2.2 Quality of Ecological Survey Data

Given a parameter θ of a population (for example, it might be the expected value, μ), suppose the purpose of the survey is to determine an estimate $\hat{\theta}$ (for example, if the parameter is the expected value μ, the estimate might be the sample mean $\bar{Y}$). There are two forms of error in estimating μ. The first, called *sampling error* (Biemer and Lyberg, 2003, p. 36), is the error due to the difference between the estimate and the parameter that results from the fact that not every member of the population is sampled. The second, called *nonsampling error* by Biemer and Lyberg (2003), refers to all of the other forms of error that can occur during the sampling process. Sampling error is an unavoidable consequence of the fact that the data represent a sample and not a census. Therefore, efforts to improve data quality are directed at nonsampling error. Biemer and Lyberg (2003) provide an extensive discussion of methods for data quality improvement.

The Moran scatterplot (Section 4.5.2) provides a very powerful general tool for assessing the quality of spatial data. Recall (Equations 4.10 and 4.11) that the Moran scatterplot computes a simple linear regression of Y, the measured variable, on WY, the spatially lagged value of Y. This permits regression diagnostics to be applied to the data in order to identify values that are discordant with their geographic neighbors. The `spdep` function `moran.plot()` employs the diagnostic statistics of the function `influence.measures()` to detect potentially discordant values. These statistics are discussed in Appendix A.2.3. Section 4.5.2 contains an example of the use of this function to identify discordant values in the detrended sand content data of Field 4.1. I am such a fan of the Moran scatterplot for this application that I am not going to discuss any alternatives, but simply present this approach in the next section.

6.2.3 Quality of Automatically Recorded Data

One of the features of modern data recording methods is that they automatically gather a very large stream of data at a high rate. Such data records warrant special attention because they often include values that would be identified as exceptional, and maybe discarded without further thought, in records collected by hand. A convenient example that typifies many of the properties of such data sets is yield monitor data in agriculture. There are a number computer programs available for quality control of yield monitor data. One excellent program is Yield Editor 2.0 (Sudduth et al., 2013). Software such as this provides an easy means to enter data directly from the yield monitor and deal with records identified as outliers or contaminants. In this section, we discuss the process of quality control in yield monitor data sets in order to develop more generally the procedure of outlier removal for automatically collected data. To give away the punch line, we are going to advocate the use of the Moran scatterplot. It is very important to emphasize that, although the discussion refers to specific events that occur in the collection of yield monitor data, analogous events occur in the collection of other forms of densely sampled point data such as remotely sensed data, soil electrical conductivity, and so forth, and the Moran scatterplot should be considered for these applications as well.

An automatically collected data stream such as that generated by a yield monitor consists of a sequence of records measured as a time series. Figure 6.1a shows a portion of the yield monitor data stream from Field 4.2. The figure shows 3,000 records out of 38,159 recorded from that field. In the general theory of outlier detection in time series, two types of outliers are commonly distinguished (Barnett and Lewis, 1994, p. 396): *additive outliers* and *innovation outliers*. An additive outlier is a "hiccup," that is, a measurement error or execution error superimposed on a single observation or set of observations and not affecting other observations outside the set. An example in combine-harvested yield monitor data is when the combine slows and turns. An innovation outlier is a measurement error or execution error that affects more than one record sequentially in the series. An example would be a change in the calibration of the sensor.

In order to properly develop the process of data quality assessment and improvement it is necessary to explicitly state one's assumptions and objective. For instance, the procedures might be very different depending on whether the objective is to discover all of the possible probability distributions underlying a data set or to exclude all data not belonging to the dominant distribution. We will assume in this section that the objective is to characterize the dominant probability distribution and to retain as many measurements of this population as we can while rejecting measurements that are suspected of containing substantial measurement error or of being members of a different population. Since we have thousands or tens of thousands of values from which to choose, we adopt the policy that any data value that can be reasonably suspected of being spurious will be deleted. This assumption is obviously very important in simplifying the analysis.

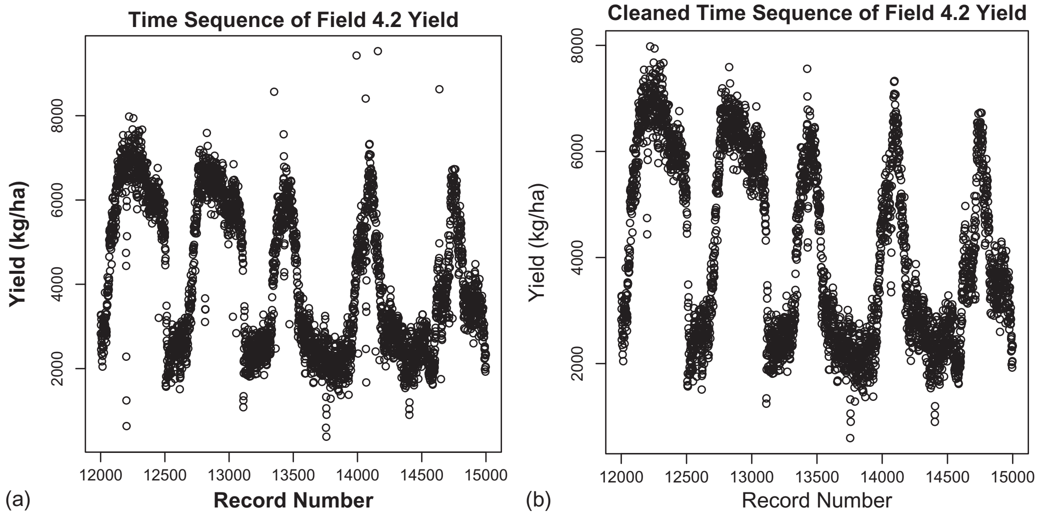

FIGURE 6.1
(a) Three thousand yield data records from Field 4.2; (b) same record interval after having eliminated data values identified as suspicious by a Moran scatterplot.

Many papers discussing elimination of outliers from data express their procedure in terms of passing the data through a series of filters, often based on the specifics of the data collection process (Edwards, 2000). In the case of yield monitor data, these filters identify and remove data satisfying criteria including the following:

1. Data recorded while or shortly after the combine header is in a raised position (Thylen and Murphy, 1996; Simbahan et al., 2004);
2. Short temporal segments of data (Simbahan et al., 2004);
3. Spatially overlapping yield records, either due to actual harvester path or to inaccurate GPS record (Blackmore and Marshall 1996; Nolan et al., 1996; Beck et al., 2001; Simbahan et al., 2004);
4. Data from less than a full swath width (Blackmore and Marshall, 1996; Blackmore and Moore, 1999; Beck et al., 2001; Colvin and Arslan, 2001);
5. Data recorded while the harvester is stopped, moving faster than a given speed, turning, accelerating, or decelerating (Murphy et al., 1995; Nolan et al., 1996; Thylen and Murphy, 1996; Blackmore and Moore, 1999; Beck et al., 2001; Colvin and Arslan, 2001; Simbahan et al., 2004);
6. Data recorded at the edge of the field (Nolan et al., 1996; Haneklaus et al., 2001);
7. Data records whose grain flow rate or moisture content are further than a given value (usually three times the standard deviation) from the mean (Beck et al., 2001; Simbahan et al., 2004);
8. Data that are outside the biological limits of the crop (Simbahan et al., 2004);
9. Data that are very different from geographically nearby values, including the results of surging in the harvester (Blackmore and Marshall 1996; Beck et al., 2001; Simbahan et al., 2004; Robinson and Metternicht, 2005).

In addition to being application specific, these criteria are somewhat ad hoc in nature. The Moran scatterplot provides a unified, theoretically justifiable means of identifying discordant values. In applying the Moran Scatterplot to yield monitor data, because we have thousands of data values and our goal is to identify the dominant probability distribution, we will discard any data records that are identified by any one of the influence diagnostics as being suspicious.

We will work with Field 4.2 from the 1996 wheat harvest to demonstrate the application of the Moran scatterplot, although we will also include code for Field 4.1. The yield data from Field 4.2 has a property that frequently occurs with automatically collected dense data sets. This is that the data include many short distance local trends, so that a value that is discordant in one region of the data set may be perfectly consistent with its neighbors in another region (Figure 6.1a).

The primary issue that arises in computing a Moran scatterplot for the yield data of Field 4.2 is the size of the data set. Since there are 38,159 data records, a spatial weights matrix relating all of them would be a 38,159 by 38,159 matrix! Although the `sp` objects such as the neighbor list and `listw` objects take advantage of the sparseness of the weights matrix (i.e., the large number of elements with value zero), a *W* matrix describing the entire Field 4.2 yield monitor data set is still beyond the reach of the computer. Accordingly, we apply the function `moran.plot()` to sub-regions sequentially. There are `sp` functions `subset.listw()` and `subset.nb()` that compute these objects for a subset of points, but for this application it appears more straightforward to use a simpler, brute force approach.

Specifically, we will subdivide the data set by Easting and Northing, apply the function `moran.plot()` to each sub-region to determine the data records that are not discordant (i.e., that we want to keep), and then reassemble these into a cleaned data set.

The raw yield data are read using code given in Appendix B.4. Since we are dealing only with point data, there is no simplification if we create an `sf` object and then coerce it, so we will use the `sp` function `coordinates()` to convert the code to a `SpatialPointsDataFrame` directly (Section 2.4.3). Because there are two fields in the data set and we would like to reuse the code for each of them, we assign the name `data.Set4` to the `SpatialPointsDataFrame` that contains the yield data. Since we will ultimately need to reassemble a group of subsets, we first assign `ID` values to make this task easier.

```
> data.Set4$ID <- 1:nrow(data.Set4)
```

Based on the rectangular shape of Field 4.2 and the capacity of the computer, we will subdivide the field into 21 sub-regions arranged in three rows of seven sub-regions per row. Field 4.1 is longer in the north–south direction, so we subdivide this field into seven rows of three sub-regions per row. First, we compute a width `x.size` and a height `y.size` for the subdivisions.

```
> # For Set 4.1, x.cells = 3 and y.cells = 7
> # For Set 4.2, x.cells = 7 and y.cells = 3
> add.factor <- 4 * Field.no - 6
> x.cells <- 5 + add.factor
> y.cells <- 5 - add.factor
> x.max <- max(coordinates(data.Set4)[,1]) + 1
> x.min <- min(coordinates(data.Set4)[,1])
> y.max <- max(coordinates(data.Set4)[,2]) + 1
> y.min <- min(coordinates(data.Set4)[,2])
```

Here we are using the function `coordinates()` as retrieval function (see Section 2.4.3). Now we need to identify each sub-region. For Field 4.2, we can number the sub-regions from 1 to 7 in the x direction and from 1 to 3 in the y direction. The following formula will compute a coordinate x_{loc} with values ranging from 1 to 7 based on the size of the x coordinate relative to the minimum and maximum x coordinate values,

$$x_{index} = trunc\left(\frac{7[x - x_{\min}]}{x_{\max} - x_{\min}}\right) + 1 \tag{6.1}$$

where the function `trunc(x)` truncates the value of `x` by removing its fractional part. Here is the implementation of this equation, placing the values computed in Equation 6.1 into a data field called `x.index`.

```
> x.index <- trunc(x.cells*(coordinates(data.Set4)[,1] -
+    x.min) / (x.max - x.min)) + 1
```

A similar operation is carried out on the y values to assign them an index of 1, 2, or 3. The next step is to combine these indices so that each of the 21 sub-regions is uniquely identified. A very simple way to do this, familiar to those who work with map algebra, is to multiply the x and y indices by different powers of 10 and then add the results of these multiplications.

```
> data.Set4$xyindex <- x.index + 10 * y.index
> print(xyindex <- unique(data.Set4$xyindex))
 [1] 31 32 33 34 35 36 37 27 17 16 15 14 13 12 11 21 22 23 24 25 26
```

The function `unique()` here returns all of the unique values of the sub-region index. Now that each sub-region is uniquely identified we can use a simple `for` loop to compute a Moran scatterplot for each sub-region. As with all irregular point sets, in creating the spatial weights matrix we must decide between using `dnearneigh()`, for distance-based neighbor definition, and `knearneigh()`, for numbers-based neighbor definition (see Section 3.6). In this case we opt for the former, since we are specifically interested in the attribute value of each data record relative to those of its nearest geographic neighbors. Some playing around is needed to get an appropriate upper distance `d2`. The code for this step is a bit long, so comments are inserted.

```
> for (i in 1:length(xyindex)){
+ # Separate out points in sub-region i
+    data.Set4.i <-
+       data.Set4[which(data.Set4$xyindex == xyindex[i]),]
+ # Compute the neighbor list and listw objects
+    nlist <- dnearneigh(data.Set4.i, d1 = 0, d2 = 20))
+    W <- nb2listw(nlist, style = "W")
+ # Compute the Moran scatterplot; don't print the results
+    mp <- moran.plot(data.Set4.i$Yield, W, quiet = TRUE)
+ # Create a matrix with 6 columns,
+ # one for each outlier identification statistic
+ # Convert the values TRUE and FALSE to numbers and keep those that
+ # are all zero (outlier = FALSE)
+    keep <- which(rowSums(matrix(as.numeric(mp$is.inf),
+       ncol = 6)) == 0)
+ # Concatenate these values to keep into one array
+    {if(i == 1)
+          keep.id <- data.Set4.i$ID[keep]
+    else
+          keep.id <- c(keep.id,data.Set4.i$ID[keep])}
+ }
```

The array `keep.id` contains the `ID` value for each of the data records that is *not* discordant (i.e., that is to be retained). First, we select those data records to keep.

```
> data.Set4.keep <- data.Set4[keep.id,]
```

These are not necessarily in numerical order, however. Therefore, we reorder them into increasing numerical order by ID value.

```
> data.Set4.keep <- data.Set4.keep[order(data.Set4.keep$ID),]
```

Let's see how many data records got dropped.

```
> nrow(data.Set4)
[1] 38159
> nrow(data.Set4.keep)
[1] 35343
```

Figure 6.1b shows the results of the cleaning on the same string of data records as is displayed in Figure 6.1a. Although a few questionable records have been retained (and possibly some good records have been discarded), overall the result is remarkably good. The final step is to aggregate the cleaned data and save them to a file in the folder *Created*. This exercise indicates the usefulness of the Moran scatterplot as a data cleaning tool. All of the analyses carried out in the remainder of the book that involve yield monitor data from one of the fields in Data Set 4 use data that have been cleaned in this way.

6.3 Spatial Interpolation Procedures

6.3.1 Inverse Weighted Distance Interpolation

One of the major difficulties associated with combining spatial data is the problem of *misalignment*. This occurs when data are collected at different locations or on different scales, and will be discussed in Section 6.4. It is evident, however, that if data measured at different locations are to be combined, then some data values will have to be predicted at locations other than those where they were measured. This is the process of interpolation, as this term is defined in geostatistics.

This section contains a brief introduction to three interpolation methods: inverse distance weighted interpolation, kriging, and co-kriging. In addition to these three methods, Thiessen polygons (Ripley, 1981, p. 38; Lo and Young, 2007, p. 333), which were discussed in Section 3.5, are often very useful for visualization via thematic maps (Cliff and Ord, 1981; Haining, 1990, p. 202), which permit the rapid visual examination of patterns in the data. For example, Thiessen polygons are used in Figure 3.9 to compare artificially generated data at differing levels of a spatial autocorrelation parameter. Thiessen polygons are also useful as a sort of last-resort interpolation method for data sets that are too sparse and irregular to use the methods described in this section.

We will formulate the pointwise interpolation problem as follows. Given a set of position coordinate vectors $\{x_1, x_2, \dots x_n\}$, where each coordinate vector x_i has two components (horizontal x and vertical y), and given a set of values $Y(x_i)$, $i = 1, 2, \dots, n$ measured at these locations, compute an interpolation of the value $Y(x)$ at a location x where the value of Y is not measured. Figure 6.2 shows a schematic example in which $n = 3$; of course in real

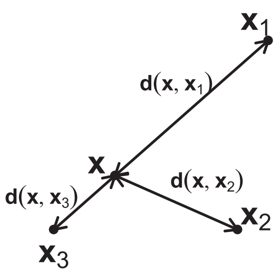

FIGURE 6.2
Schematic representation of the location of three points x_1, x_2 and x_3 at which Y is measured, and a fourth location x, at which the value of Y is to be estimated.

situations n will generally be larger than this. An interpolator $\hat{Y}(x)$ is called a *linear interpolator* if for some set of coefficients ϕ_i, $i = 1, 2, \ldots, n$,

$$\hat{Y}(x) = \sum_{i=1}^{n} \phi_i Y(x_i). \tag{6.2}$$

If the values $Y(x_i)$ are spatially uncorrelated, then their spatial location is irrelevant and the best (in a sense that will be made clear below) interpolator is the mean $\hat{Y}(x) = \bar{Y}$, independent of x. If, however, the values of $Y(x_i)$ are positively spatially autocorrelated, then we can expect that values of $Y(x_i)$ measured at locations close to x will be closer to the value of $Y(x)$ than values farther away. For example, in Figure 6.2, one would expect that the value of $Y(x_3)$ will be closer to that of $Y(x)$ than will the value of $Y(x_1)$, with $Y(x_2)$ somewhere in between. Thus, one would expect that increasing the value of ϕ_3 in relation to ϕ_1 and ϕ_2 in Equation 6.2 would in this case produce a more accurate interpolation.

At this point, we can note one very important property of the ϕ_i. Since the values $Y(x_i)$ in Equation 6.2 are random variables, the interpolator $\hat{Y}(x)$ is a random variable as well. An interpolator $\hat{Y}(x)$ of $Y(x)$ is *unbiased* if its expected value $E\{\hat{Y}(x_i)\}$ is equal to $Y(x)$ (Isaaks and Srivastava, 1989, p. 234). If we consider the case of uncorrelated Y values, and interpolate using the mean $\bar{Y} = \Sigma Y_i / n$, we can see that in this case $\phi_i = 1/n$ for all n, and therefore

$$\sum_{i=1}^{n} \phi_i = 1. \tag{6.3}$$

The mean is unbiased. This provides intuition for a property that turns out to be true in general, which is that in order for the interpolator $\hat{Y}(x)$ to be unbiased, the ϕ_i must satisfy Equation 6.3 (Isaaks and Srivastava, 1989, p. 234).

Inverse distance weighted (IDW) interpolation (Isaaks and Srivastava, 1989, p. 257; Lo and Yeung, 2007, p. 350) is a simple, *ad hoc* method that sometimes produces reasonably good results. Returning to Figure 6.2 for intuition, the interpolator $\hat{Y}(x)$ can be improved relative to the sample mean by reducing the weight of more distant locations like x_1 relative to closer ones like x_3, that is, by reducing the value of ϕ_i for locations with large distance values $d(x, x_i)$. We can achieve this by making ϕ_i proportional to $d(x, x_i)^{-p}$ for some $p \geq 1$. We need to scale the ϕ_i so that Equation 6.3 is satisfied, and we can do this by dividing each value of $d(x, x_i)^{-p}$ by their sum $\Sigma_{i=1}^{n} d(x, x_i)^{-p}$. This leads to the formula for the inverse distance weighted method,

$$\begin{aligned} \hat{Y}(x) &= \sum_{i=1}^{n} \phi_i Y(x_i)., \\ \phi_i &= \frac{d(x, x_i)^{-p}}{\sum_{i=1}^{n} d(x, x_i)^{-p}}. \end{aligned} \tag{6.4}$$

There are two "control parameters" in this equation, the power p and the number of neighbors n. Varying these parameters changes the interpolator $\hat{Y}(x)$. Increasing the value of

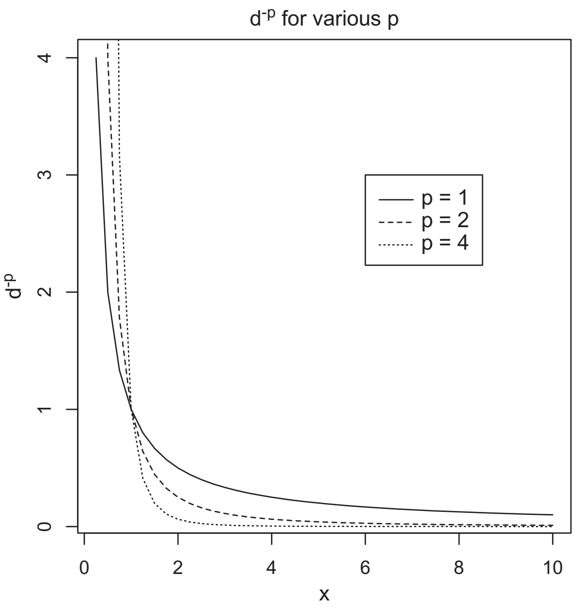

FIGURE 6.3
Curves representing d^{-p} as a function of d for three values of p. These indicate the relative rate at which the influence of data on an interpolated value drops off as a function of distance from the sampled location to the interpolated location.

n increases the number of values that contribute to the interpolation. Some interpolation software also permits the analyst to specify a maximum geographic distance $d(x, x_i)$ beyond which a value $Y(x_i)$ is not included. Figure 6.3 shows graphs of the function d^{-p} for p equal to 1, 2, and 4. Increasing the value of p reduces the influence of more distant neighbors relative to that of closer ones. The "standard" value of p in most interpolation software is 2.

In Chapter 5, we used an artificial population with 10,368 values to compare various sampling methods. We will use that same population in this section to compare interpolation methods. Using the function `closest.point()` from Chapter 5, the artificial yield data are sampled at the same 78 locations in Field 4.2 at which environmental data were sampled (these are shown in Figure 5.7).

```
> sample.coords <- cbind(data.Set4.2$Easting, data.Set4.2$Northing)
> # Find the yield at the closest sample points
> samp.pts <- apply(sample.coords, 1, closest.point,
+    grid.data = pop.data)
> data.Set4.2$Yield <- pop.data$Yield[samp.pts]
> coordinates(data.Set4.2) <- c("Easting", "Northing")
```

There are several packages in R that do IDW interpolation. We will use the package `gstat` (Pebesma, 2004), which links nicely with `sp` objects. First, we must create a grid of points at which to interpolate the data. We will interpolate on a 5-meter grid one-half meter inside the sample boundary.

```
> print(Left <- bbox(bdry.spdf)[1,1])
[1] 592105
> print(Right <- bbox(bdry.spdf)[1,2])
[1] 592815
> print(Top <- bbox(bdry.spdf)[2,2])
[1] 4267858
> print(Bottom <- bbox(bdry.spdf)[2,1])
[1] 4267503
> cell.size <- 5
> grid.xy <- expand.grid(Easting = seq(Left,Right,cell.size),
+   Northing = seq(Top,Bottom,-cell.size))
```

We first convert `grid.xy` into a `SpatialPoints` object and then into a `SpatialGrid` object

```
> coordinates(grid.xy) <- c("Easting", "Northing")
> gridded(grid.xy) = TRUE
```

The actual IDW interpolation can be done with the `gstat` function `idw()`. This is an R *wrapper* function, in other words, a function that performs a specialized task by calling another function that performs a more general task. In this case, the more general function is `predict.gstat()`(Rossiter, 2007). The purpose of wrapper functions is to simplify programming by making it easier to specify the arguments of the function.

```
> yield.idw <- idw(Yield ~ 1, data.Set4.2, grid.xy, idp = 2, nmax = 12)
```

The first argument is a `formula` that specifies that `Yield` is being interpolated, the second argument gives the `SpatialPointsDataFrame`, and the third and fourth arguments specify the tuning parameters p and n in Equation 6.4. Note! If you get the following message

```
Error: is.ppp(X) && is.marked(X) is not TRUE
```

it means that you have executed `library(spatstat)` and have this package, which also has an `idw()` function, hanging around. In this case, execute

```
> yield.idw <- gstat::idw(Yield ~ 1, data.Set4.2, grid.xy,
+   idp = 2, nmax = 12)
```

Figure 6.4 shows gray-scale maps of interpolated yields computed using four combinations of n and p in Equation 6.4. Figure 6.4a shows the results of what are generally considered "typical" values, $n = 12$ and $p = 2$. The round spots surrounding certain sample points, commonly called "duck eggs," are an indication that the value of $Y(x_i)$ at that x_i is sufficiently extreme to produce a "bulge" or a "hollow" in the interpolated surface. Reducing the value of n to 4 (Figure 6.4b) reduces the duck egg effect in this data set but does not eliminate it. Raising the value of p to 4 (Figure 6.4c) strongly emphasizes the nearest data location. As p becomes larger, the influence of the nearest location increases (cf. Figure 6.3), so that for large p the thematic map begins to resemble a set of Thiessen polygons. Reducing the value of p to 1 has the opposite effect (Figure 6.4d). The increased influence of more distant locations has a smoothing effect on the interpolation.

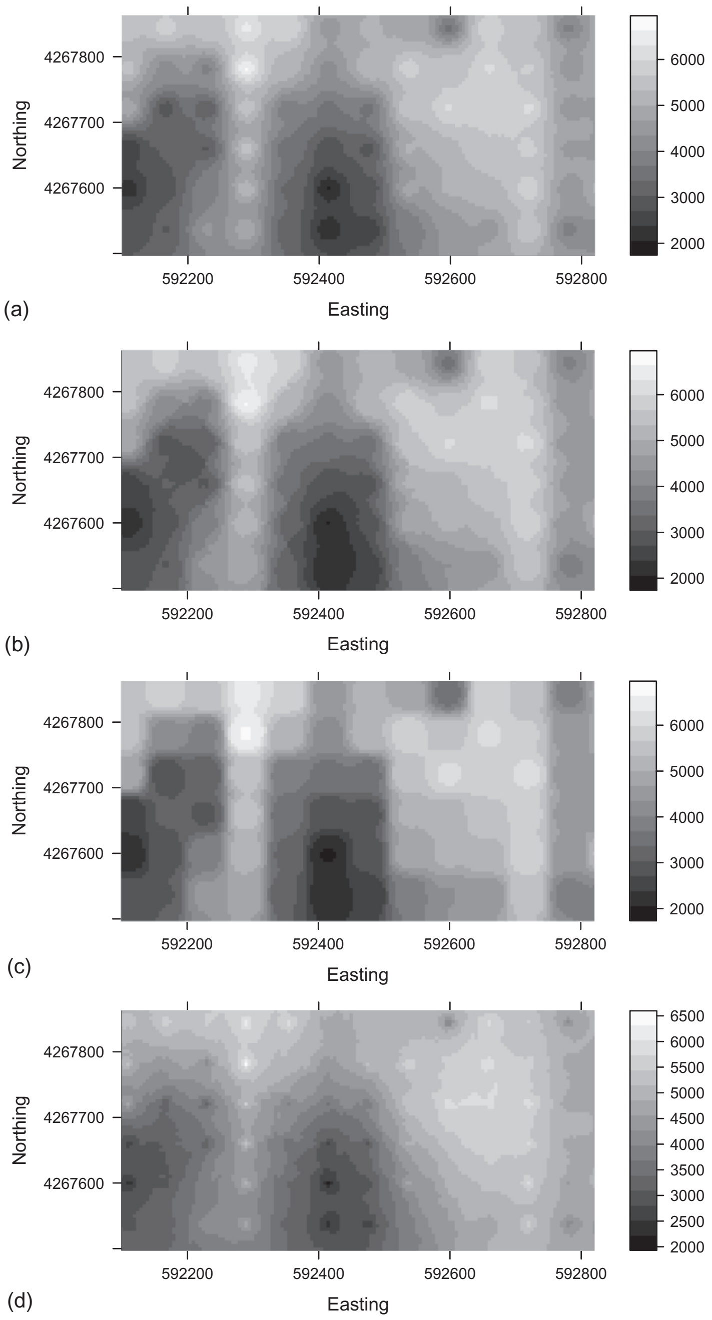

FIGURE 6.4
Thematic maps of IDW interpolation of data sampled from the artificial data set at 78 sample locations. The maps show the effect of varying the number of neighbors *n* and the power value *p* in Equation 6.4 (a) Field 4.2 interpolated yield, n = 12, p = 2; (b) Field 4.2 interpolated yield, n = 4, p = 2; (c) Field 4.2 interpolated yield, n = 12, p = 4; and (d) Field 4.2 interpolated yield, n = 12, p = 1.

6.3.2 Kriging Interpolation

The term "kriging" was originally used by Georges Matheron (Matheron, 1963) in honor of the South African mining engineer D.G. Krige, who carried out early work on this method (Krige, 1951). Those in the know, by the way, say that "kriging" is pronounced with a hard "g," like "get." There are several varieties of kriging, and we will restrict our discussion to ordinary kriging. The ordinary kriging interpolator $\hat{Y}(x)$ is a linear interpolation, which means that it is defined by Equation 6.2, similarly to the IDW interpolator. IDW interpolation is an ad hoc calculation based on the intuitive idea that the influence of a value $Y(x_i)$ should die off as the distance between x and x_i increases. Kriging interpolation is based on the assumption of a probability model for $Y(x)$, and is derived within that model with the objective of minimizing the variance of the interpolator $\hat{Y}(x)$. We will assume that Y is second-order stationary (this assumption makes things simpler but is not strictly necessary (see Cressie, 1991, p. 142).

A derivation of the equations of the ordinary kriging estimator is given by Isaaks and Srivastava (1989, p. 281). The error variance is given by

$$\operatorname{var}\{\hat{Y}(x) - Y(x)\} = \operatorname{var}\{\sum_{i=1}^{n} \phi_i Y(x_i) - Y(x)\}. \tag{6.5}$$

Using the formula for the variance of a linear combination of random variables, Equation 6.5 can, after some algebra, be written (Isaaks and Srivastava, 1989, p. 284)

$$\begin{aligned}\operatorname{var}\{\hat{Y}(x) - Y(x)\} = \hat{\sigma}^2 + \sum_{i=1}^{n}\sum_{j=1}^{n} \phi_i \phi_j \operatorname{cov}\{Y(x_i), Y(x_j)\} \\ -2\sum_{i=1}^{n} \phi_i \operatorname{cov}\{Y(x_i), Y(x)\}.\end{aligned} \tag{6.6}$$

To minimize the error variance, one sets to zero the derivatives of the right side of Equation 6.6 with respect to each of the ϕ_i, subject to the constraint $\Sigma\phi_i = 1$ to ensure that the interpolation is unbiased (Equation 6.3). This is a constrained optimization problem and is solved by the method of Lagrange multipliers (Appendix A.4). We introduce a Lagrange multiplier ψ so that the quantity to be minimized becomes

$$\begin{aligned}F(\phi, \psi) = \hat{\sigma}^2 + \sum_{i=1}^{n}\sum_{j=1}^{n} \phi_i \phi_j \operatorname{cov}\{Y(x_i), Y(x_j)\} \\ -2\sum_{i=1}^{n} \phi_i \operatorname{cov}\{Y(x_i), Y(x)\} + \psi\left(\sum_{i=1}^{n} \phi_i - 1\right).\end{aligned} \tag{6.7}$$

The solution is obtained by taking the partial derivatives with respect to ψ and each of the ϕ_i and setting them to zero. The result is (Isaaks and Srivastava, 1989, p. 287)

$$\sum_{j=1}^{n} \phi_j \operatorname{cov}\{Y(x_i), Y(x_j)\} + \psi = \operatorname{cov}\{Y(x_i), Y(x)\},\ i = 1,\dots,n$$

$$\sum_{i=1}^{n} \phi_i = 1. \tag{6.8}$$

At this point, one must introduce a model for the covariance functions in Equation 6.8. Assuming that the data are isotropic, we can write $\operatorname{cov}\{Y(x_i), Y(x_j)\} \cong \tilde{C}(h)$, where $\tilde{C}(h)$ is a model for the covariance defined in Equation 4.24 between two values separated by a distance h, and h is the distance between x_i and x_j. Although we could use the covariogram model directly, it is traditional in geostatistics to define the covariogram in terms of the variogram model. Let $\hat{\gamma}(h)$ be the experimental variogram of the data set, defined in Equation 4.21, and let $\tilde{\gamma}(h)$ be a variogram model, for example, the spherical model defined in Equation 4.22. Let us denote the nugget by $\tilde{\gamma}_0$ and the sill by $\tilde{\gamma}_\infty$. Let $\tilde{C}(h)$ denote the corresponding model for the covariogram. Then from Equation 4.26 it follows that

$$\tilde{C}(h) = \tilde{\gamma}_\infty - \tilde{\gamma}_0 - \tilde{\gamma}(h). \tag{6.9}$$

In words, the covariogram model is the sill minus the nugget minus the variogram model.

The ordinary kriging procedure is to fit the data with an experimental variogram, use Equation 6.9 to compute a covariogram model for the data, and then substitute this model into the *kriging equations*

$$\sum_{j=1}^{n} \phi_j \tilde{C}_{ij} + \psi = \tilde{C}_{i0},\ i = 1,\dots,n$$

$$\sum_{i=1}^{n} \phi_i = 1, \tag{6.10}$$

where $\tilde{C}_{ij}$ is the value of the covariogram model at a lag h corresponding to the distance between x_i and x_j, and $\tilde{C}_{i0}$ is the covariogram model at a lag h corresponding to the distance between x_i and x. These values of ϕ_i are then substituted into Equation 6.2 to obtain $\hat{Y}(x)$.

The first step in kriging interpolation is to compute the experimental variogram and fit it with an appropriate variogram model. Using the same data as in Section 6.3.1, we can accomplish these two steps using the `gstat` functions `variogram()` and `fit.variogram()`.

```
> data.var <- variogram(Yield ~ 1, data.Set4.2, cutoff = 600)
> data.fit <- fit.variogram(data.var, model = vgm(150000, "Sph", 250,
+   1))
```

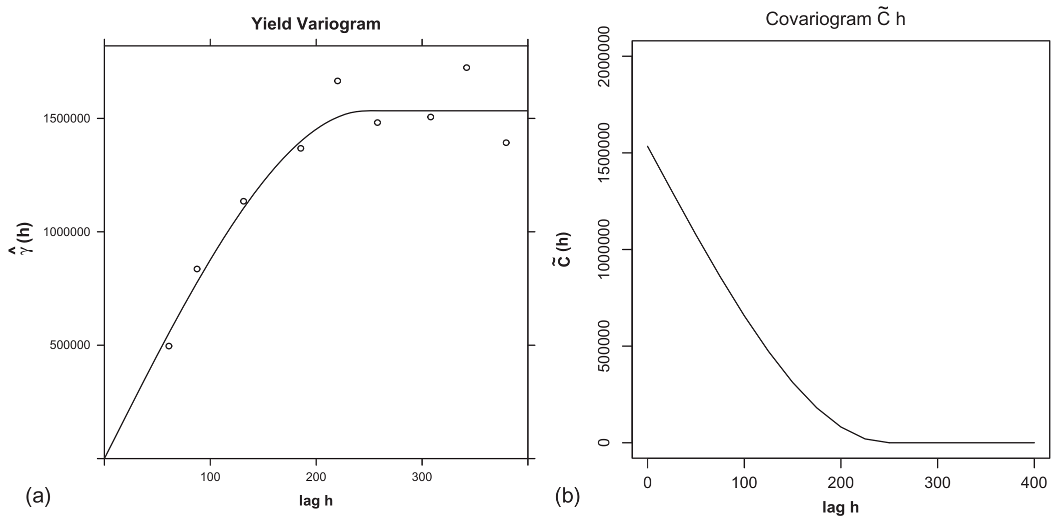

FIGURE 6.5
(a) Variogram of yield data based on 78 sample points; (b) covariogram of the same yield data as part (a).

Figure 6.5a shows the experimental variogram for the artificial yield data at the 78 sample points, together with the best fit spherical variogram model. Without going into a lot of detail, the spherical model is the one that works most often. The arguments of the function `vgm()` inside the function `fit.variogram()` are respectively, the starting estimate for the sill, the variogram model, the starting estimate for the range, and the starting estimate for the nugget. In actual practice, these were obtained from examination of the plot of the experimental variogram, but for the purposes of drawing the figure in the code the function `fit.variogram()` is run prior to generating the plot.

The nugget is zero, the range is 248.28m, and the sill is 1,533,695 (kg/ha)^2. Since in our example the nugget is zero, the covariogram model is given by the sill minus the variogram model. This model is shown in Figure 6.5b. The covariogram model behaves in a similar manner to an inverse distance (Figure 6.3), with both of them tapering off to zero as the lag distance h increases. Thus, the terms $\tilde{C}_{i0}$ on the right-hand side of Equations 6.10 provide the same distance attenuation effect as the inverse distances in the IDW method. With isotropic data we have $\tilde{C}_{i0} = \tilde{C}(h)$, where h is the lag distance between point x and point x_i. The distance modeled by $\tilde{C}(h)$ is sometimes called the *statistical distance.* The role played by the terms $\tilde{C}_{ij}$ on the left-hand side of the equation is the distinguishing feature of kriging (Isaaks and Srivastava, 1989, p. 300). If two data locations are close together, they will tend to have large $\tilde{C}_{ij}$ values. When the Equations 6.10 are solved for the ϕ_i, the effect of these large $\tilde{C}_{ij}$ values is to adjust the raw statistical distances measured by $\tilde{C}_{i0}$ to take into account the effects of redundancy due to close proximity of sample locations.

Kriging is carried out in the `gstat` package using the wrapper function `krige()`. The first three arguments are the same as those of the function `idw()` in the previous section. The fourth argument specifies the variogram model and indeed, if this argument is not specified then `krige()` will compute an IDW interpolation.

```
> yield.krig <- krige(Yield ~ 1, data.Set4.2, grid.xy,
+   model = data.fit)
```

Figure 6.6 shows a gray scale plot of the kriged artificial yield data.

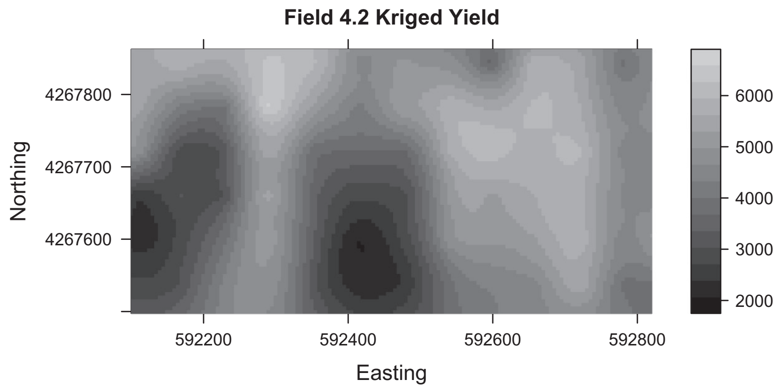

FIGURE 6.6
Thematic map of kriging interpolation of data sampled from 78 sample locations in the artificial data set.

It is important to note that although kriging provides the best linear unbiased interpolation of the data, in the sense that at any given point the interpolation has the highest precision, kriging does not provide a very good model of the *variability* of the data. Indeed, kriging, like any interpolation method, tends to smooth the data, that is, to reduce the variance of the distribution (Ripley, 1981, p. 51; Isaaks and Srivastava, 1989, p. 268). This effect can be visualized by comparing the kriged interpolation (Figure 6.6) with the original data (Figure 5.1a). Simulation methods such as Gaussian simulation (Ripley, 1987; Deutsch and Journel, 1992) are much better suited for generating a model that simulates the variability of the data. In Exercise 6.7, you are asked to compare the errors associated with IDW and kriging interpolation. The results may surprise you a bit.

6.3.3 Cokriging Interpolation

It is often the case that when trying to interpolate the values of one quantity that is expensive to measure, one also has access to data on one or more related quantities that are relatively inexpensive to measure. Consider the example of measuring clay content in Field 4.2. To obtain these data, one goes to the field with a soil coring tool and at each sample location one extracts about five or so soil cores from locations within a radius of a few meters. One then combines these into a single soil sample and takes it to the lab for analysis. There are various ways of determining sand, silt, and clay content, but all are sufficiently expensive to deter high intensity sampling. In non-saline soils, apparent soil electrical conductivity (EC_a) is related to clay content (Triantafilis et al., 2001; Corwin and Lesch, 2005; Sudduth et al., 2005). This is cheaper and easier to measure, and therefore can be sampled more intensively. Cokriging provides a means by which the correlated EC_a measurements can be used to augment the clay content measurements to obtain a more accurate interpolation.

A high density set of soil EC_a measurements is available in the file *Set4.2EC.csv*. In addition to measurement locations in WGS84 and UTM coordinates, this file contains two data fields, *ECto30* and *ECto100*. These correspond to the horizontal and vertical dipole measurements of the instrument and represent the approximate depth in centimeters to which EC_a is measured. The contents of this file are loaded into the data frame `data.Set4.2EC` (Appendix B.4), which is then converted into a `SpatialPointsDataFrame`. Figure 6.7 shows the measurement locations of the EC_a sample locations as well as the

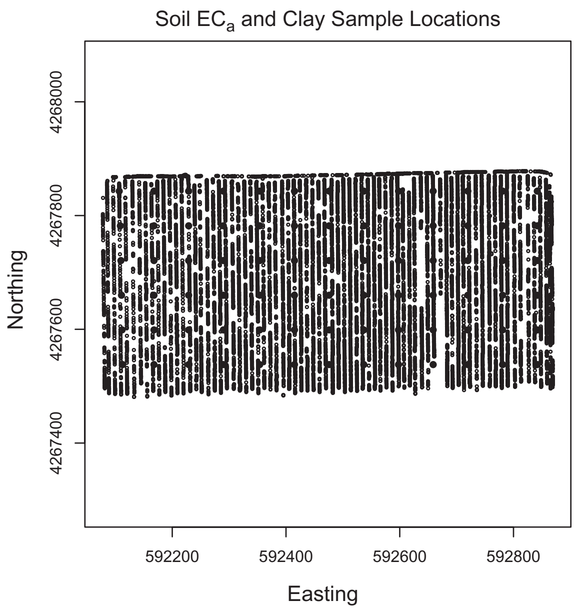

FIGURE 6.7
Sample locations of soil EC_a (small open circles) and clay content (large filled circles) for Field 4.2.

78 soil sample locations. The contents of the file *Set4.296sample.csv* are loaded into the data frame `data.Set4.2` and this is converted into a `SpatialPointsDataFrame`. By applying the function `closest.point()` developed in Chapter 5 to EC_a, we can compare the two EC_a measurements with clay content.

```
> EC.pts <-
+    apply(sample.coords, 1, closest.point, grid.data = data.Set4.2.EC)
> data.Set4.2@data$ECto30 <- data.EC@data$ECto30[EC.pts]
> data.Set4.2@data$ECto100 <- data.EC@data$ECto100[EC.pts]
```

Figure 6.8 shows a plot of clay content vs. *ECto30*, which turns out to have the closer linear relationship with clay content ($r = 0.53$). We will use this as the covariate in our cokriging interpolation of soil clay content.

The code to make Figure 6.8 is as follows.

```
> with(data.Set4.2@data, plot(ECto30, Clay, cex.main = 2,
+    main = expression(Soil~EC[a]~vs.~Clay~Content),
+    xlab = expression(EC[a]~"("*dS/m*")"), cex.lab = 1.5,
+    ylab = "Percent Clay")) # Fig. 6.8
```

There are two possibly discordant points in Figure 6.8, and this provides an opportunity to introduce the function `identify()`. This function takes as arguments the values on the abscissa and ordinate of the scatterplot and a value to use as a label. After invoking this function, you can use the cursor to click on a point, and it will be identified by the label. Here is the code to label the discordant points.

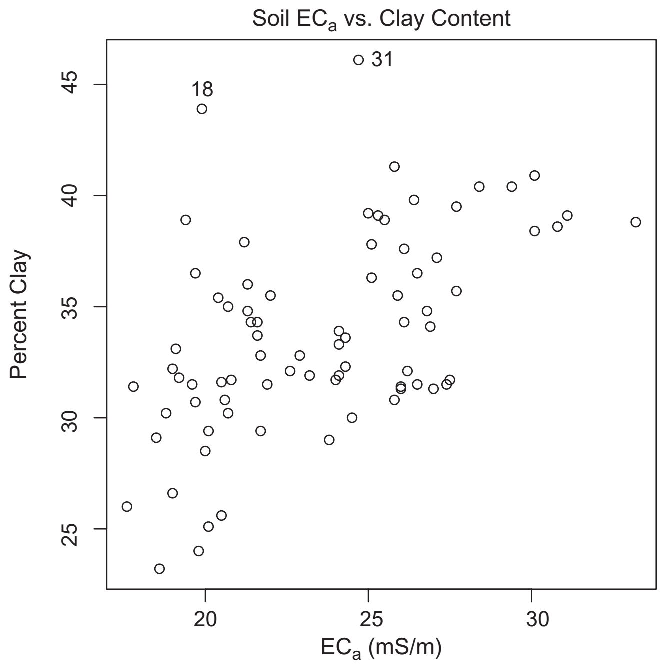

FIGURE 6.8
Scatterplot of the relationship of clay content to the variable *ECto30*, representing electrical conductivity measured to approximately 30 cm. Two possibly discordant points have been identified.

```
> identify(data.Set4.2@data$ECto30, data.Set4.2@data$Clay,
+     data.Set4.2@data$ID)
```

If you are using RStudio, you won't see anything until you click on the *Finish* button in the upper right corner of the *Plot/Help* window (after first clicking on the two discordant points). Points 18 and 31 are identified as the discordant values, so we will try removing them.

```
> data.Set4.2.cleaned <- data.Set4.2[-c(18,31),]
```

This improves the r value to 0.61, so we will go with the cleaned data.

The derivation of the cokriging equations is similar to that of the kriging equations given in the previous section, and therefore is not discussed here. The interested reader is referred to the excellent exposition in Isaaks and Srivastava (1989, Chapter 17). IDW and kriging interpolation can be implemented in the `gstat` package through the use of wrapper functions, but cokriging requires the use of the fundamental function `predict.gstat()`. As with the implementation of function polymorphism in R generally, you do not have to actually specify this function name in an R statement in order to use it. Rather, you specify the function `predict()` with an object from the `gstat` package as the first argument, and R automatically applies `predict.gstat()`. The first step is therefore to construct a `gstat` object. This is done using two applications of the `gstat()` constructor function.

```
> g.cok <- gstat(NULL, "Clay", Clay ~ 1, data.Set4.2)
> g.cok <- gstat(g.cok, "ECto30", ECto30 ~ 1, data.Set4.2.EC)
```

Cokriging in gstat requires that both the variate and the covariate have the same range. Therefore, we construct and inspect the experimental variograms of these two quantities, using the functions variogram() and fit.variogram() as with kriging. The range of *Clay* is about 400m and the range of *ECto30* is about 300m (variogram plots not shown), so we will us a range of 350m for the cokriging interpolation of the cokriging object g.cok.

```
> g.var <- variogram(g.cok)
> g.fit <- fit.lmc(g.var, g.cok, vgm(1, "Sph", 350, 1))
```

Now we are ready to carry out the cokriging interpolation. This is done a call to the polymorphic function predict().

```
> g.cokrige <- predict(g.fit, grid.xy)
```

After staring at the screen for a few minutes, we decide to go out and get a cup of coffee. Upon our return (if you actually try this, it may take several hours), we find that the computer has good news and bad news for us. First, the good news.

```
Linear Model of Coregionalization found. Good.
[using ordinary cokriging]
```

The word "Good" refers to the fact that the model g.fit can indeed be used for cokriging, because the ranges have been matched correctly. Now for the bad news.

```
There were 50 or more warnings (use warnings() to see the first 50)
> warnings()
Warning messages:
1: In predict.gstat(g.fit, grid.xy):
   Covariance matrix singular at location [592100,4.26786e+006,0]:
skipping...
```

Problems like this are often caused by numerical overflow when the computer chokes on too much data. One way to determine whether that is the problem in this case is to try cokriging the clay content data in Field 4.2 by using the soil EC_a data from that same data set. You are asked to do this in Exercise 6.8. It turns out that this is indeed the problem, so we will reduce the size of the covariate data set by using the modulo function to eliminate a fraction of the data records. If you did Exercise 2.6, you learned about this function. For those who did not, the *modulo* operator u mod v, for integer values of u and v, is the remainder of the division of u by v. Thus, for example, 6 mod 5 = 1. The modulo operator in R is %% so that the operation u %% v evaluates to u mod v. If we number the data records in data.EC from 1 to n, and select only those records whose record number i satisfies i mod $k = 0$ we will eliminate $(k-1)/k$ of the records.

```
> nrow(data.Set4.2.EC)
[1] 12002
```

With 12,000 data records, we can probably eliminate nine-tenths of them and still have a useful covariate. We will reload the data file and carry out this operation.

```
> data.EC <- read.csv("Set4\\Set4.2EC.csv")
> ID <- 1:nrow(data.EC)
> mod.fac <- 10
> data.ECmod <- data.EC[which(ID %% mod.fac == 0),]
> coordinates(data.ECmod) <- c("Easting", "Northing")
```

This executes without error. Let's see what we get.

```
> g.cok <- gstat(NULL, "Clay", Clay ~ 1, data.Set4.2)
> g.cok <- gstat(g.cok, "ECto30", ECto30 ~ 1, data.ECmod)
> g.var <- variogram(g.cok)
> g.fit <- fit.lmc(g.var, g.cok, vgm(1, "Sph", 350, 1))
> g.cokrige <- predict(g.fit, grid.xy) # This works
Linear Model of Coregionalization found. Good.
[using ordinary cokriging]

> names(g.cokrige)
[1] "Clay.pred"        "Clay.var"         "ECto30.pred"      "ECto30.var"
[5] "cov.Clay.ECto30"
```

Figure 6.9 shows the cokriged and kriged clay data. They are obviously similar but not identical. Unlike the kriged artificial yield population data, we do not have a clay content population that we can use to compare the two methods. We can, however, make use of cross validation. We will discuss this method in detail in Chapter 9 in connection with

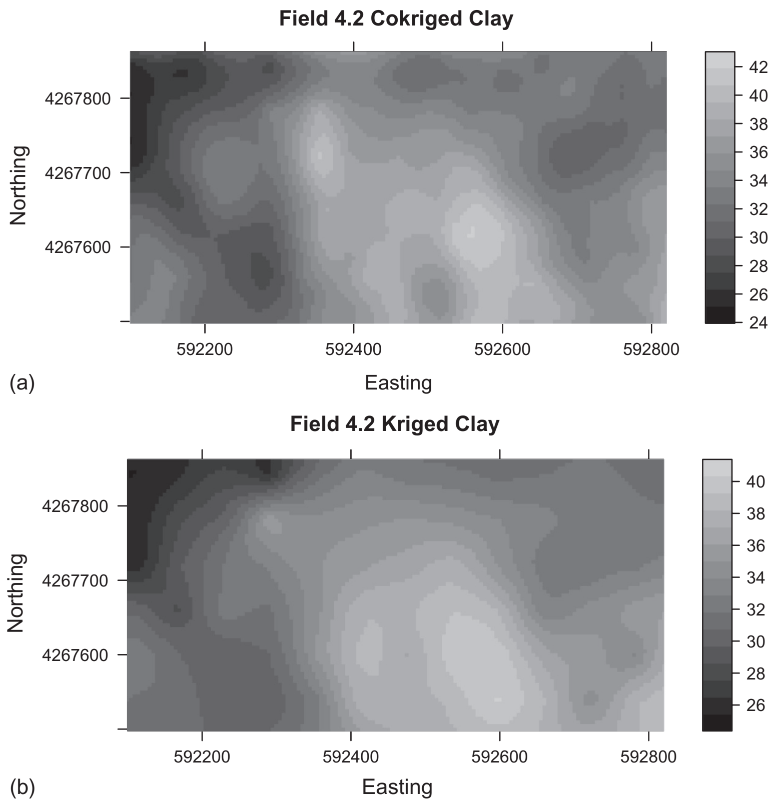

FIGURE 6.9
(a) Thematic map of clay content interpolated using ordinary kriging. (b) Thematic map of clay content interpolated using cokriging with EC_a.

generalized additive models and recursive partitioning. The basic idea is very simple. The *k-fold cross validation* procedure is to remove $1/k$ of the data records from the data set, compute the predicted values at the locations of these removed data records, and compare the predictions with the actual values. The process is repeated for successive fractions $1/k$ of the records until all records in the data set have been subjected to the procedure. One can then compute diagnostic statistics. Two of the most useful are the mean error, which indicates the bias, and root mean square error (RMSE), which indicates the variance (Section 6.2.2).

For kriging, the `gstat` library includes the wrapper function `krige.cv()`. In the application to kriging and cokriging, the most common practice is to remove and predict the data records one at a time.

```
> claykrige.cv <- krige.cv(Clay ~ 1, data.Set4.2, grid.xy,
+    model = clay.fit, nfold = nrow(data.Set4.2))
> mean(res.krige) # Mean error (bias)
[1] -0.005734252
> sqrt(mean(res.krige^2)) # RMSE (variance)
[1] 2.89842
```

In the code above, the argument `nfold = nrow(data.Set4.2)` indicates one at a time cross validation. As with cokriging itself, cokriging cross validation does not have a wrapper function. In some ways, however, the code is even easier because it uses the `gstat` object `g.cok`.

```
> claycok.cv <- gstat.cv(g.fit, nfold = nrow(data.Set4.2))
> res.cok <- claycok.cv$residual
> mean(res.cok)
[1] -0.02391489
> sqrt(mean(res.cok^2))
[1] 2.769164
```

The mean error of the cokriging residuals in this example is actually larger, while the RMSE is slightly smaller.

6.4 Spatial Rectification and Alignment of Data

6.4.1 Definitions of Scale Related Processes

Adjustments in the spatial component of a data set generally involve three primary operations. The first is georegistration, which is the alignment of the coordinates defining the spatial component of a data record with the true geographic coordinates of the corresponding location of the spatial entity. Georegistration is easiest to do in a GIS, and we will not discuss it here. The second adjustment is to transform the projection if necessary. This process can be carried out in R and is described in Section 2.4.3. The third operation, which is the focus of this section, is the correction of spatial misalignment. We will make use of the interpolation techniques discussed in Section 6.3 to resolve misalignment in spatial data. The discussion is organized in terms of the theory of scale change developed by Bierkens et al. (2000). In order to discuss concepts involving scale precisely, it is necessary to define terms precisely. Unfortunately, much of the terminology is not standardized, and indeed the term *scale* has a second meaning in cartography that is the reverse of the one used here (Lo and Yeung, 2007, p. 22). To standardize our own terminology,

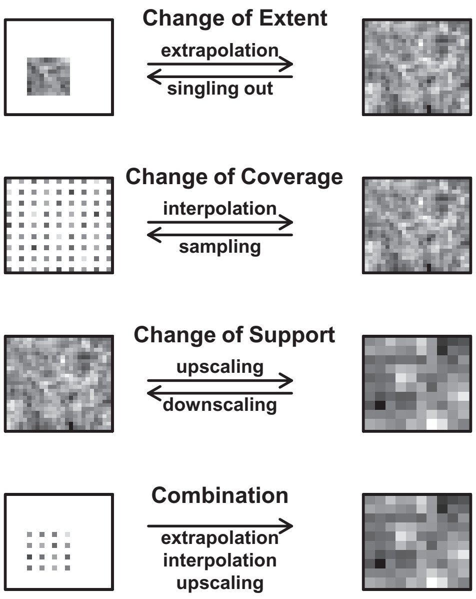

FIGURE 6.10
Schematic map of the three scale change operations defined by Bierkens et al. (2000). Redrawn from Figure 5.5 of Bierkens et al (2000, p. 9). Original figure copyright 2000, Kluwer Publishing. Used with kind permission from Springer Science and Business Media B.V.

we will use that of Bierkens et al. (2000). All of our definitions are quoted directly from their glossary on pp. 177–179. They define three scale change operations shown schematically in Figure 6.10. These are change of extent, change of coverage, and change of support.

The *extent* is the "area…over which observations are made, model outcomes are calculated, or policy measures are to be made." Suppose we make measurements in one of the rice fields of Data Set 3 and use them to represent the whole data set. This process, "increasing the extent of the research area," is called *extrapolation*. Conversely, if we use measurements made over the whole region to draw conclusions about a single field, this process of "selecting an area of smaller extent" is called *singling out*.

The term *support* is defined as "the largest…area…for which the property of interest in considered homogeneous." For example, the sand content data shown in Figure 3.4 were collected by taking four or five soil cores over an area of about 5 m^2 and compositing these cores. Thus, the support of these data is about 5 m^2. The term *scale* as we use it here is synonymous with support, with *support* being the commonly used term in the geostatistics literature. The process of moving from a smaller support (or scale) to a larger one, so that one is "increasing the support of the research area," is called *upscaling*. The opposite process, that of "decreasing the support of the research area," is called *downscaling*.

The third class of change of scale operations involves the *coverage*. Referring to the data describing Field 4.2, pixels values from an aerial image are available at any location in the field, with a support of about 3 m^2. The point sample data, on the other hand, were measured at 78 locations in the field, each with a support of about 5 m^2. The soil content value is known only at these locations, whose total support is about 400 m^2 in a 32 ha field. The ratio of "the sum of all the support units for which the average values are known and the extent" is called the *coverage*. If we want to estimate the value of, say, clay content at a location not sampled, we must interpolate it. The precise definition of *interpolation* is analogous to the geostatistical definition: the process of "increasing the coverage of the research area." The process of "selecting and observing a subset" of the support units in the extent is called *sampling*, matching the colloquial use of the term. The term *change of scale* is used somewhat loosely to refer generally to a change of one or more of these three features, support, extent, and coverage.

It should be noted that the definition given by Bierkens et al. (2000) of the term *support* leaves out one important feature. The definition should include not only the size, but also the shape of the area over which the phenomenon in question is averaged (Gotway and Young, 2002). For example, a lattice with a given number of square cells has the same average area as an irregular mosaic with the same number of cells, but their different spatial configurations may result in different values of statistics describing them, so that changing the configuration as well as the size of the subdivisions of a spatial region should be considered a change of support, even though it is neither upscaling nor downscaling.

We will not discuss in this book operations involving a change of extent. The process of extrapolation can only be meaningfully carried out in the presence of covariates, and the question of how to carry it out correctly is primarily a scientific one rather than a statistical one. The process of singling out primarily a GIS operation and is referred to in GIS terminology as "clipping," and can be done either using a GIS or functions in the package `rgeos` (Bivand and Rundel, 2017). The next two subsections contain a brief discussion of the two remaining operations, change of support, and change of coverage, as used in data set alignment.

6.4.2 Change of Coverage

One of the two change of coverage operations, sampling, is the topic of Chapter 5. The second change of coverage operation is interpolation. One of the primary concerns with the use of geostatistical interpolation to predict data at unmeasured locations is the effect of this operation on the distributional properties of the resultant data set. In exploratory analysis one is most concerned with developing data sets that can be used to visualize patterns in the data using maps and other tools of data exploration. Thus, there is little need to be concerned with the effect of the operations on the distributional properties of the data. Things are very different when one is preparing the data for parameter estimation and hypothesis testing (the phase that Tukey, 1977, p. 3, calls confirmatory analysis). In this case, changes in the distributional properties of the data induced by operations such as interpolation can have a profound effect on the statistical analysis (Anselin, 2001). This is true in particular of the smoothing effect of interpolation mentioned in Section 6.3.2.

Consider the problem of fitting a linear regression model of the form $Y = X\beta + \varepsilon$, where for example Y is a vector of crop yield values based on yield monitor data and X is a matrix of explanatory variables from data collected at sample locations. Figure 6.11 shows three forms of spatial data describing a small portion of Field 4.2. Yield is measured by a yield monitor at the locations indicated by the open circles, soil EC_a is measured at the locations indicated by the "+" signs, manual sampling was carried out at the location indicated by the * symbols, and the pixels of a remotely sensed image cover the areas indicated by the shades of gray. If one were to use IDW or kriging interpolation to predict all of these data values on a common grid, the interpolation would induce error into explanatory variables as well as into the yield data. A crucial assumption of the theory of linear regression is that the explanatory variables are measured without error (Sprent, 1969; Webster, 1997). Violations of this assumption must be handled by the *errors in variables* theory of linear regression (Johnston and DiNardo, 1997, p. 153).

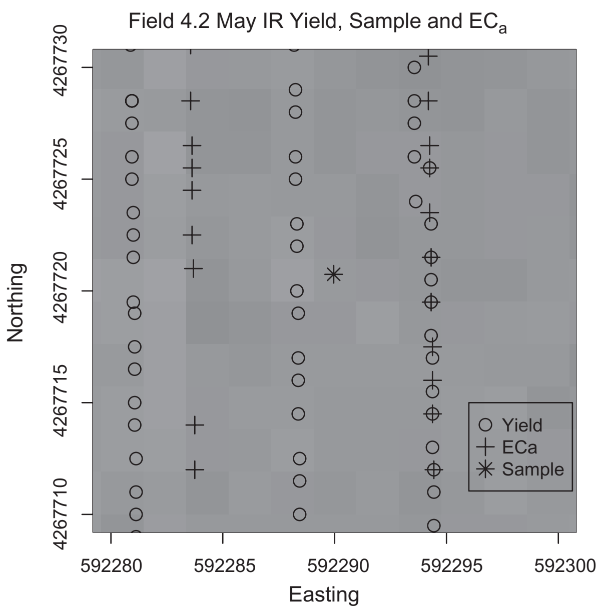

FIGURE 6.11
Spatial arrangement of the data sets from Field 2 of Data Set 4. The asterix represents a sampling location, the plus signs represent electrical conductivity measurements, the circles represent 1996 yield monitor locations, and the gray squares are pixels in the infrared band of a remotely sensed image taken in May 1996.

Fortunately, in determining the effects of environmental variables on an automatically measured response variable (such as yield in the application we discuss here, where the X values are sample data and the Y values come from a yield monitor), it is reasonable to assume that the environmental variables are measured more accurately than the response variable. Therefore, the assumptions of exact measurement of the explanatory variables required by the regression model can be considered to be satisfied if the value of the response variable is estimated at the spatial locations of the explanatory variables. In our analyses of Data Set 4, we will generally interpolate all variables to the locations of the manual sample sites. Griffin et al. (2007) provide an excellent practical discussion of protocols to follow in carrying out this procedure. One way to do this is to create buffers around each sample point and estimate the yield at each sample point as the mean of those yield monitor values falling within the buffer circle. This is called the *neighborhood mean* and is analogous to the compositing operation commonly practiced when taking soil samples. As with the size of the region over which to composite a soil sample, the size of the buffer circles is somewhat arbitrary. We will use a value of 10m so that each buffer would enclose some data values from at least two and possibly three passes of the combine harvester.

The data are loaded as described in Appendix B.4. The buffering operation can be carried out using the `spatstat` (Baddeley and Turner, 2005) function `disc()` or the `rgeos` function `gBuffer()`. We will use the former. This creates a circular `owin` object (this is a form of `spatstat` object whose details we don't need to know). We can create a disc of radius 10m centered on first sample location as follows.

```
> xy <- coordinates(data.Set4.2)[1,]
> samp.rad <- 10
> d <- disc(radius = samp.rad, centre = xy)
```

Note the spelling of the argument `centre`. We don't need to know the details of the `owin` object because we immediately coerce it into a `SpatialPolygons` object.

```
> buffer.10m <- as(d, "SpatialPolygons")
```

The function `spRbind()` joins pairs of polygons, so we will augment the object `buffer.10m` via a `for` loop that creates and adds new buffer discs one at a time.

```
> for (i in 2:nrow(data.Set4.2@data)){
+    xy <- coordinates(data.Set4.2)[i,]
+    d <- disc(radius = samp.rad, centre = xy)
+    sp.d <- as(d, "SpatialPolygons")
+    sp.d@polygons[[1]]@ID <- as.character(i)
+    buffer.10m <- spRbind(buffer.10m, sp.d)
+ }
```

The attribute data come from a different source from the spatial data, so as always we must be careful to give each disc the correct ID number before adding it. These sorts of comparisons are generally most easily done with spatial features object because of their simpler structure.

```
> library(sf)
> data.Set4.2.sf <- as(data.Set4.2, "sf")
> buffer.10m.sf <- as(buffer.10m, "sf")
> st_crs(buffer.10m.sf) <- "+proj=utm +zone=10 +ellps=WGS84"
> y <- st_contains(buffer.10m.sf, data.Set4.2.sf)
> unlist(y)
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
[22] 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42
[43] 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
[64] 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78
> all.equal(data.Set4.2.sf$ID, unlist(y))
[1] TRUE
```

Figure 6.12a shows a small portion of the field with four buffer circles.

The computation of the means over the buffer circle of each of the data fields in the yield monitor data file can be carried out using the R function `over()` to determine the mean value of those data records in the yield file `data. Yield4.2` that lie inside each buffer circle.

```
> yield.bufmean <- over(buffer.10m, data.Set4.2Yield96raw, fn = mean)
```

This usage of the function `over()` is as follows. The objects `buffer.10m` and `data.Set4.2Yield96raw` are `SpatialPolygons` and `SpatialPointsDataFrame`, respectively. In this case, the function `over()` returns a data frame with the same number of elements (length of a vector or rows of a data frame) as the first argument. If the third argument is not specified, `over()` returns the index of the polygon. We want to compute the mean yield over each buffer, and we accomplish this by specifying `fn = mean` as the third argument, which applies the function `mean()` over the data locations lying within each polygon in `buffer.10m`. It is possible, although it takes more work, to compute the same mean values with the order of arguments of `over()` reversed (Exercise 6.10). Since the buffers are circular, the means of the geographic coordinates will be near but generally

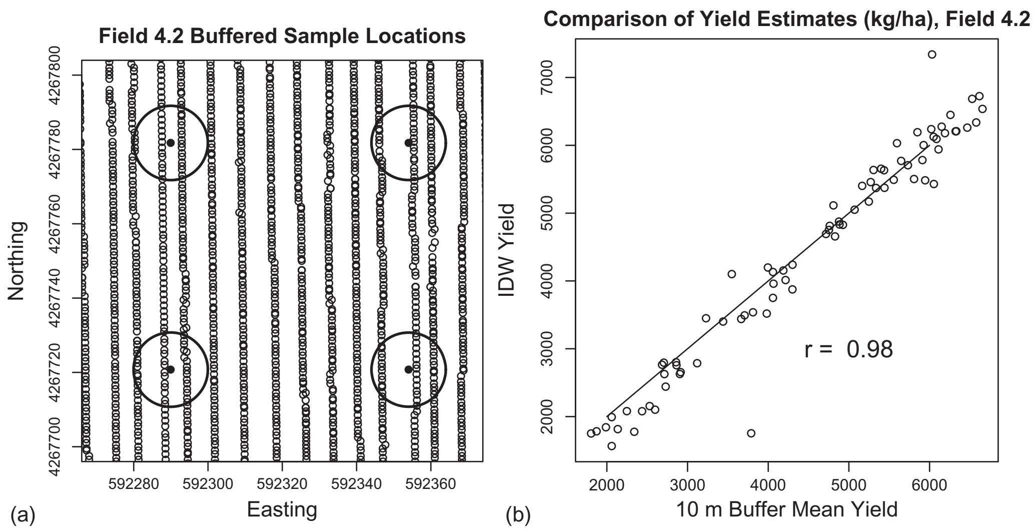

FIGURE 6.12
(a) Yield monitor sampling points of Field 4.2 in 1996 (white circles), sampling points of sparse data (black circles), and 10m buffers of the sparse sample points (large circles); (b) Scatterplot of the yield values obtained via the two methods. The solid line satisfied $y = x$.

not exactly equal to the geographic coordinates of the sample points, which are at the center of the buffer circles, but they should be close enough.

Given the issues with the effect of interpolation on the distribution of the interpolated data, it is of interest to compare the yield estimates in Field 4.2 obtained using the 10m buffers of Figure 6.12a with those obtained using interpolation. Figure 6.12b shows a plot of the two estimates. The correlation between the two estimates is $r = 0.99$. This indicates that for this data set the operation of aggregating data over a region of radius 10 meters has approximately the same smoothing effect on the estimation process as does the operation of IDW interpolation. This is not necessarily true, however, for all data sets. As always, the best procedure is to test both methods, breathe a sigh of relief, and move on if they give similar results, or try to find out what is going on if they do not. For our later use, we will create a file *yield4.296ptsidw* holding the interpolated values of yield and grain moisture and put it in the *created* folder.

```
> yield.idw.df <- data.frame(Yield = yield.idw$var1.pred,
+    Easting = coordinates(yield.idw)[,1],
+    Northing = coordinates(yield.idw)[,2])
+ write.csv(yield.idw.df, "created\\set4.2yld96ptsidw.csv")
```

In Exercise 6.10, you are asked to repeat this process for Field 1. The file you create will be important for subsequent analysis.

6.4.3 Change of Support

The process of change of support (see Figure 6.10) via interpolation is known in the GIS literature as *resampling*. Resampling for exploratory analysis involves two primary considerations. The first is how to interpolate Y values at a common set of points so that data values may best be visualized in the process of data exploration. This requires the aggregation of values over a common lattice or interpolating values at a common set of points. The second consideration involves the proper way to aggregate different data sets so that they may be compared and incorporated into statistical models. We will defer discussion of this second issue until Section 11.5, and for now focus on the issue of data exploration.

Since spatial data sets may contain thousands or tens of thousands of points, the standard forms of visualization, both spatial (e.g., thematic maps) and non-spatial (e.g., boxplots, scatterplots, etc.) may be virtually illegible unless the data are summarized to some extent. For this reason, a common practice is to predict the Y values at a set of locations having a larger support than the data. This can be done by aggregating over a mosaic of polygons or interpolating to a common set of data points. Aggregation over a mosaic can be carried out, for example, by computing the mean over each polygon of the points within it. The data may in principle be interpolated using any interpolation method, including IDW interpolation and kriging as covered in Sections 6.3.2 and 6.3.3 (Nolan et al., 1996). Where no well-defined mosaic exists a priori, we will focus on the second option, interpolation to a common grid and then sampling at a subset of grid locations.

There is, in particular, a considerable literature discussing the interpolation of yield monitor data to a regular grid, and this may serve as a model for interpolation of other similar data types. Most of this work involves a comparison between kriging and IDW interpolation. Because of the relatively large sample size and the relatively small distance between samples, one would not expect a great difference between results using different interpolation methods (Birrell et al., 1995). Indeed, comparisons between different forms of interpolation indicate little difference (Juerschik et al., 1999; Robinson and Metternicht, 2005). Kriging is more complex but has the advantage that it provides a best linear unbiased interpolation of the data at each point as well as a variance estimate (Birrell et al., 1995; Blackmore and Marshall, 1996; Thylen and Murphy, 1996; Blackmoore and Moore, 1999). The data smoothing property of interpolation methods may be problematic for statistical modeling and confirmatory analysis, but it may actually be useful for exploratory analysis since it may be possible that important patterns obscured by high data variability are more visible when this variability (i.e., noise) is reduced.

Unlike automatically sampled data sets, remotely sensed data generally consist of a pixilated grid. Each pixel contains a set of integer values (often three values in the case of multispectral sensing, and many more in hyperspectral sensing) that represent the intensity of radiation in different bands of the electromagnetic spectrum. For example, in the common case of a false color infrared image these are infrared (represented as red), red (represented as green), and green (represented as blue). If the ultimate objective is to sample image data on a subset of grid cells, it is often not necessary to place remotely sensed images onto a common grid to do so. If it is necessary to convert remotely sensed image data, or any other form of pixilated data, to a common grid, one simple operation to do so is nearest neighbor resampling (Lo and Young, 2007, p. 154).

Figure 6.13a shows a close-up view of a small portion of the remotely sensed image of Field 4.2 taken in May 1996. The "+" symbols are located at the center of each pixel and represent the location at which the band values of that pixel are considered to be correctly measured. The black dots represent the locations at which the values are to be resampled. Each pixel can be considered as a Thiessen polygon surrounding its center point, so that assigning to the resample points the value of the pixel in which they are located accomplishes nearest neighbor resampling. The sp function `over()` can be used to carry out this nearest neighbor resampling operation, as can the `raster` function `resample()`. Since we used `over()` in the last section, we will use `raster()` in this one. Assume that the image is loaded as the R object `data.Set4.2.May` (Appendix B.4) and that values of the edges of

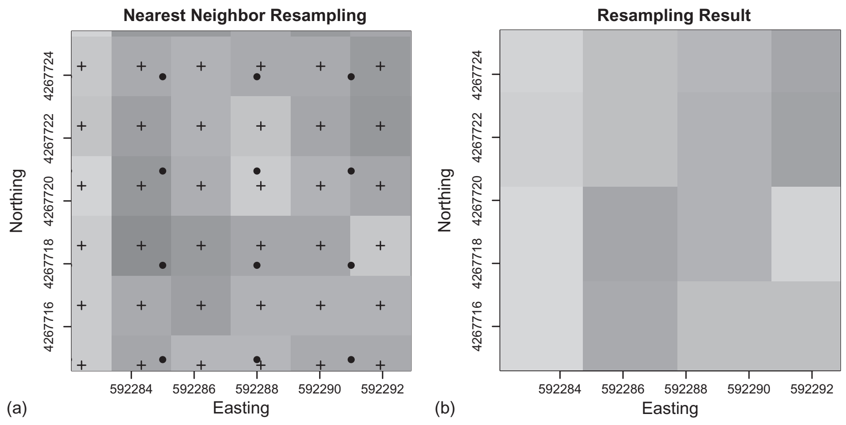

FIGURE 6.13
(a) Close-up view of a small portion of the remotely sensed image of Field 4.2 taken in May 1996. The "+" symbols are located at the center of each pixel and represent the location at which the band values of that pixel are assumed to be correctly measured. The black dots represent the locations at which the values are to be resampled. Each black dot is assigned the value of the plus sign contained in the same pixel. (b) Gray squares indicating the results of nearest neighbor resampling on the data of part (a).

the field have been assigned to R objects `N`, `S`, `E`, and `W`. The first step is to use the function `raster()` to read the existing image and generate a cell grid at the new cell size.

```
> library(raster)
> data.Set4.2.May <- raster("set4\\Set4.20596.tif")
> proj4string(data.Set4.2.May) <- CRS("+proj=utm +zone=10
+ellps=WGS84")
> cell.size <- 1.9
>
```

Next, we create a new raster object `new.ras` that defines the grid size of the resampled data.

```
> E <- 591967
> W <- 593139
> S <- 4267140.95
> N <- 4268320.95
> new.cell.size <- 3
>
> ncol.new <- (W - E) / new.cell.size
> nrow.new <- (N - S) / new.cell.size
> new.ras <- raster(ncol = ncol.new, nrow = nrow.new, xmn = E,
+   xmx = W, ymn = S, ymx = N)
> proj4string(new.ras) <- CRS("+proj=utm +zone=10 + ellps=WGS84")
```

Finally, the function `resample()` is used to resample the raster `data.Set4.2.May` to the new grid defined by `new.ras`, creating a raster object `data.Set4.2.new`.

```
> > data.Set4.2.new <- resample(data.Set4.2.May, new.ras, method =
+   "ngb")
```

The third argument specifies that nearest neighbor resampling will be used. This is discussed below. Figure 6.13b shows the results of this resampling. On my computer, neither of the plots in Figure 6.13 could be produced in RStudio because of memory limitations, but they did work in TINN-R.

In resampling from a raster grid, such as a remotely sensed image, the usual assumption is that the precise location of the value of the raster cell (pixel) is at its center. Based on the discussion in this chapter, there are at least three possible ways that one could develop a common alignment of estimated values from a set of densely measured data that includes two or more quantities. These are (1) a nearest neighbor resampling scheme to a common grid in which the estimated values at the aligned points are set to the geographically nearest data value; (2) a local aggregation scheme in which, for example, Thiessen polygons are constructed around each of the aligned points and the estimated value is set at the mean of those data values falling within the corresponding polygon; and (3) an interpolation scheme such as kriging in which values at aligned locations are interpolated from the data. The distribution of the data created from each of these methods will be different, and some evaluation method will be necessary to compare them. The raster function `resample()` provides two methods to choose from, bilinear interpolation (the default) and nearest neighbor resampling.

The cell size in the new grid in the example just provided is about one-and-one-half times the size of that of the original grid. Hijmans (2016) points out that for substantial changes in the size of the support (i.e., the cell size in this case), one of the raster functions `aggregate()` or `interpolate()` should be used to more accurately determine the appropriate attribute values of the resampled data.

There are some very serious statistical issues that must be dealt with. These have been alluded to previously, but let us gather some of them in one location. First, these dense data sets are highly autocorrelated in space, so that assumptions of independence of errors generally do not hold. Second, the relationship between, for example, remotely sensed image data and a response variable can be highly dependent on the support over which the data are aggregated (Long, 1996). This phenomenon is called the *modifiable areal unit problem* (covered in Section 11.5.1). Third, if one is using the image data to predict the response variable value, there may be substantial measurement error in both the predictor variable (e.g., image data) and the response variable, in violation of the assumptions of linear regression. Fourth, the action of georegistration, in which features in an image are brought into alignment with their geographic locations, can affect the distributional properties of the data, which compromises the conclusions of a statistical analysis (Anselin, 2001). We will not attempt to address these issues right now, but it will be necessary to keep them in mind in future chapters.

6.5 Further Reading

In addition to Lo and Yeung (2007), which we use as the primary GIS reference, there are many others that provide excellent discussions of issues such as map projections, Thiessen polygons, spatial resampling, and so forth. Prominent among these are de Smith et al. (2007), Bonham-Carter (1994), Burrough and McDonnell (1998), Clarke (1990), and Johnston (1998). Good GIS texts at a more introductory level include Chang (2008), Demers (2009), and Longley et al. (2001).

There is some literature on data quality in ecological surveys (e.g., Kanciruk et al., 1986; Edwards, 2000; Iverson, 2007). The literature on sample surveys carried out via interviews is much more extensive, however, and many of the concepts carry over directly to ecological sampling. A classic is Kish (1965). At a slightly lower level mathematically is Biemer and Lyberg (2003). Naus (1975) discusses some of the issues associated with automatic error detection and data adjustment for large data sets. Anderson et al. (1990) provide an interesting discussion of the data quality issues associated with a large, well-known study, the Framingham Heart Study. Naus (1975) provides more general information about the maintenance of database quality.

Wickham and Grolemund (2017) describe the *tidyverse*, a collection of R packages for the input, graphing, cleaning, and analysis of data sets. A data set is said to be *tidy* if it can be arranged in a set of tables whose columns represent the fields, such that within every field each record forms a row. In that context, our data sets are tidy. The graphics package `ggplot2` is one part of the tidyverse, but there are several other parts as well. Small data sets such as ours that are conveniently packaged in csv files and images may not need the tidyverse, but if your data set is a mass of notes and miscellaneous files collected without too much care for how they fit together, then you may want to look into this.

Barnett and Lewis (1994) is an excellent source for outlier detection and treatment. The papers by Anscombe (1960), Daniel (1960), and Kruskal (1960) are a part of a volume on the detection and treatment of discordant values and, despite their age, they still contain a wealth of useful knowledge. This chapter has focused on the identification of individual outliers. For a discussion of the identification of multiple outliers, see Davies and Gather (1993).

Burrough and McDonnell (1998) provide a discussion of the effect of scale on quality of spatial data. Bierkens et al (2000) provide a very systematic, almost recipe-like approach to changes of scale. There have not been very many implementations of their approach; for one example see Hauselt and Plant (2010). The effect of scale changes is greatest when the interactions among the variables are nonlinear. For discussions in an agricultural context see Cassman and Plant (1992) and Lark and Webster (2001). Odeh et al. (1998) provide an excellent example of the use of information at multiple scales to direct sampling. There is a vast literature on hierarchy theory, which provides organizing principles for spatial and temporal scales in ecosystems. The usual entry to this is O'Neill et al. (1986).

Isaaks and Srivastava (1989) is the standard introductory text for geostatistics. Journel and Huijbregts (1978) and Goovaerts (1997) provide excellent discussions that go beyond the material in Isaaks and Srivastatva (1989). Cressie (1991) is an excellent source at a higher mathematical level. Kanevski and Maignam (2004) provide a good discussion of interpolation methods, stochastic simulation, as well as neural network and support vector machine analysis, for environmental data.

Kutner et al. (2005) is a good introductory source for regression diagnostics. Belsley et al. (1980) and Fox (1991, 1997) provide a more extensive discussion.

Exercises

6.1 The square-shaped northern set of sample points in Data Set 2 is the Klamath Range. Create a map of this region, and display the map along with a map of the boundary of California, in both longitude-latitude and UTM coordinates. The region is in UTM Zone 10N.

6.2 The data field *QUDO_BA* in Data Set 2 contains measurements of the total basal area of blue oak trees at each sample location. Use the function `moran.plot()` to identify discordant values of this variable and create a map showing their location. Can you distinguish any possible source of the error based on the map?

6.3 (a) Use the function `expand.grid()` to create a rectangular grid of sample points in the (x, y) domain in which x takes on values 0, 10, 20, and 30, and y takes on the values 20, 10, and 0. Use the function `class()` to verify that this grid is a data frame. Create a third data field called `z` in this data frame and assign it numbers beginning with 1 and incrementing by 1. Load the package `maptools`. Use the function `coordinates()` to assign `x` and `y` as the coordinates of the data frame. Repeat the application of the function `class()` to the object. (b) Display the grid using the function `plot()`. Set the argument `pch` in the function `plot()` to the `z` data field. Carry out this same operation plotting the value of `z` using the function `spplot()`. (c) Use the function `gridded()` to convert the object to a `SpatialPixelsDataFrame`. Repeat the applications of `plot()` and `spplot()` of part (b).

6.4 Construct a set of Thiessen polygons built around the 78 data points of Field 2 of Data Set 4. Display the map in R along with the axes in UTM coordinates, using yield from the artificial population to determine the gray scale. Based on the discussion in Section 6.3.1, what is an example of a data set for which Thiessen polygons would be the most appropriate?

6.5 Use IDW interpolation to interpolate the yield data of Field 4.1 to the sample points and save the file as set4.1yldptsidw.csv.

6.6 The `sp` package contains the function `over()` that performs a GIS overlay operation. This can be used to clip an interpolated grid to an irregular boundary. If you have not yet created the boundary file for Field 4.1 by doing Exercise 2.11, create it now. Compute an IDW interpolation of *Clay* on a 5m grid. Apply `over()` to clip the interpolated grid to the field boundary. What class of object is the result? Use this result to create a `SpatialPointsDataFrame` of the interpolated clay content with the correct boundary. Use `spplot()` to verify your result.

6.7 The clay content data of Field 1 of Data Set 4 has a strong north–south trend, which violates the stationarity assumption. Prepare two kriging interpolations of clay content, one of which involves detrended data and other of which does not detrend the data, and compare the two. Don't forget to add the trend back onto the kriged detrended data. Notice that a spherical model will not work very well for the non-detrended data, so you will have to try something else.

6.8 Carry out IDW and kriging interpolation of the artificial yield population on the same grid as that population and compute the errors. (a) Plot the errors using `spplot()`. How do they compare visually? (b) Generate histograms of the distributions of the two error sets. (c) Compute the sums of squared errors of the two sets. (d) Can you think of circumstances under which IDW would be preferable to kriging?

6.9 The file *data, Set4.2.csv* contains two data fields representing soil EC_a: *EM38F*, the soil EC_a measured in the furrow, and *EM38B*, the soil EC_a measured in the beds. Determine which correlates better with soil clay content, and use this to carry out a cokriging interpolation of clay content with EC_a as the covariate. Compare your result with that obtained by ordinary kriging.

6.10 Repeat the computation of yield means over records located in each circular neighborhood in Figure 6.12a, but do it with a reversed order of arguments in the function `over()`.

6.11 Repeat for Field 1 of Data Set 4 the creation of a file *set4.1yld96ptsidw.csv* analogous to the one created in Section 6.4.2 doe Field 2.