4

Analyzing Geostatistical Data

This chapter introduces functions available in S-PLUS and S+SPATIALSTATS for analyzing geostatistical data. Geostatistical data, also termed random field data, consist of measurements taken at fixed locations. For a complete description of geostatistical data see chapter 1. Specifically, this chapter discusses methods related to variogram analysis and kriging. Variogram estimation and kriging were originally introduced as geostatistical methods for use in mining applications. In recent years, these methods have been applied to many disciplines including meteorology, forestry, agriculture, cartography, climatology, and fisheries.

In this chapter you will learn about the following topics:

- Estimating Variograms (section 4.1).
- Fitting Theoretical Variogram Models (section 4.2).
- Performing Ordinary and Universal Kriging (section 4.3).
- Simulating Geostatistical Data (section 4.4).

4.1 Variogram Estimation

Geostatistical data typically exhibit small-scale variation that may be modeled as spatial autocorrelation and incorporated into estimation procedures. The variogram provides a measure of spatial correlation by describing how sample data are related with distance and direction. In general, two closely

neighboring data are more likely to have similar values than two data farther apart. Other distance-based measures of spatial correlation include the correlogram and covariogram functions.

In this section we continue exploratory data analysis for geostatistical data by looking at tools that can be used to generate empirical variograms. Variogram estimation is exploratory and is often a multi-step process; a final variogram model is constructed from knowledge of the underlying processes affecting (generating) the data and by customization of the available tools.

S+SPATIALSTATS provides a variety of functions for use in variogram estimation. The subsections below describe tools for calculating empirical variograms, generating variogram clouds, detecting and removing trend, and exploring and correcting for anisotropy. Variogram analysis is not necessarily performed following this linear sequence of topics; the final description will likely be based on an iterative sequence of analyses using a combination of the possible tools. Your starting point for variogram analysis and the order in which you choose to proceed, will depend on how much you already know about your data. For example, if you know that your data contains a trend, then your analysis might begin by modeling the trend (see section 4.1.3).

4.1.1 The Empirical Variogram

The empirical variogram provides a description of how the data are related (correlated) with distance. The semivariogram function, $\gamma(h)$, was originally defined by Matheron (1963) as half the average squared difference between points separated by a distance h. The semivariogram is calculated as

$$\gamma(h) = \frac{1}{2|N(h)|} \sum_{N(h)} (z_i - z_j)^2$$

where $N(h)$ is the set of all pairwise Euclidean distances $i-j = h$, $|N(h)|$ is the number of distinct pairs in $N(h)$, and z_i and z_j are data values at spatial locations i and j, respectively. In this formulation, h represents a distance measure with magnitude only. Sometimes, it might be desirable to consider direction in addition to distance. In such cases, h will be represented as the vector $\mathbf{h}$, having both magnitude and direction.

Note: The terms semivariogram and variogram are often used interchangeably. By definition, $\gamma(h)$ is the semivariogram and the variogram is $2\gamma(h)$. For conciseness, however, this manual will refer to $\gamma(h)$ as the variogram.

The main goal of a variogram analysis is to construct a variogram that best estimates the autocorrelation structure of the underlying stochastic

process. Most variograms are defined through several parameters; namely, the *nugget effect*, *sill*, and *range*. These parameters are depicted on the generic variogram shown in figure 4.1 and are defined as follows:

- *nugget effect*—represents micro-scale variation or measurement error. It is estimated from the empirical variogram as the value of $\gamma(h)$ for $h = 0$.
- *sill*—the $\lim_{h \to \infty} \gamma(h)$ representing the variance of the random field.
- *range*—the distance (if any) at which data are no longer autocorrelated.

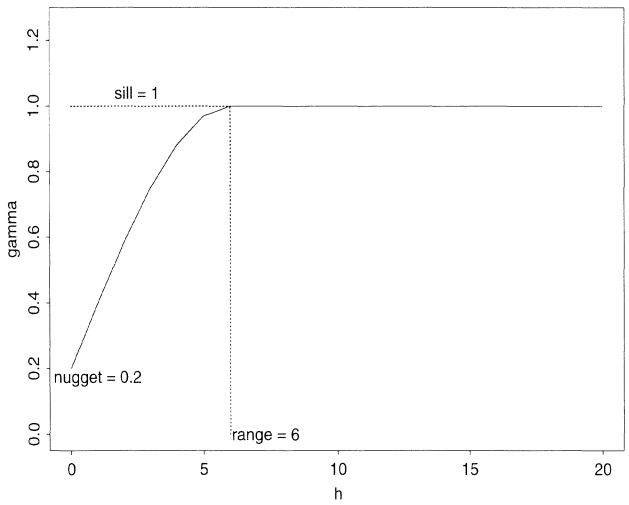

FIGURE 4.1. A generic variogram showing the *sill*, and *range* parameters along with a *nugget effect*.

Construction of a variogram requires consideration of the following:

- an appropriate *lag increment* for h;
- a *tolerance* for the lag increment; and
- the *number of lags* over which the variogram will be calculated.

The *lag increment* defines the distances at which the variogram is calculated. The *tolerance* establishes distance bins for the lag increments, to accommodate unevenly spaced observations. The *number of lags* in conjunction with the size of the lag increment will define the total distance over which a variogram is calculated.

There are two practical rules (Journel and Huijbregts, 1978) that should be considered when making your choices for lag increment and number of lags:

1. The experimental variogram should only be considered for distances h for which the number of pairs is greater than 30.
2. The *distance of reliability* for an experimental variogram is $h < D/2$, where D is the maximum distance over the field of data.

Omnidirectional Variograms

One way to begin a variogram analysis is to compute the omnidirectional variogram. An omnidirectional variogram is computed without respect to direction; all possible directions are combined into a single variogram. Use the S+SpatialStats function `variogram` to generate the onmidirectional variogram for the coal ash data :

```
> coal.var1 <- variogram(coal ~ loc(x,y), data=coal.ash)
```

The `variogram` function requires a formula defining the response in terms of the spatial locations. The `loc` function is used to incorporate the location variables as predictors, and to distinguish them from the usual S-Plus model specification for predictor variables. The `variogram` function also has a number of arguments that may be set to customize the variogram. These include `lag`, `nlag`, `tol.lag`, `maxdist`, and `minpairs`. By default, `variogram` yields an omnidirectional variogram with `maxdist` equal to the distance of reliability. The number of lags, `nlag`, is set to 20. The lag increment, `lag`, is automatically calculated as `maxdist/nlag`. The lag tolerance, `tol.lag`, defaults to `lag/2`.

The `variogram` function returns an object of class `"variogram"`. The components of `coal.var1` are the `distance`, $\gamma(h)$ (`gamma`), the number of pairs (`np`), and the `azimuth`:

```
> coal.var1
    distance    gamma   np azimuth
 1  1.201634 1.202911  719       0
 2  2.000000 1.172307  331       0
 3  2.236068 1.321759  644       0
 4  3.036036 1.314383 1170       0
 5  3.605551 1.297816  545       0
 6  4.234139 1.398687 1518       0
 7  5.039712 1.531598 1142       0
 8  5.472889 1.537340  638       0
```

```
 9  6.063483 1.536710 1382       0
10  6.552482 1.624369  719       0
11  7.147503 1.478895 1307       0
12  7.894487 1.491406 1269       0
13  8.459312 1.654917  882       0
14  9.094676 1.714728 1012       0
15  9.624306 1.804034  685       0
16 10.174499 1.656996  995       0
17 10.843929 1.705829  682       0
18 11.400797 1.880180  860       0
```

The calculated distance is the average distance between all pairs of points in the given distance bin. The azimuth is the clockwise angle from north, in degrees, defining the direction in which the variogram is calculated. The default onmidirectional variogram is based on `azimuth` = 0. To view a summary of the call to `variogram`:

```
> summary(coal.var1)
Call:
variogram.formula(formula = coal ~ loc(x, y),
                  data = coal.ash)
       lag nlag  maxdist
 0.6041523   20 12.08305


    distance          gamma            np
 Min.   : 1.202   Min.   :1.172   Min.   : 331.0
 1st Qu.: 3.763   1st Qu.:1.341   1st Qu.: 682.8
 Median : 6.308   Median :1.534   Median : 871.0
 Mean   : 6.338   Mean   :1.518   Mean   : 916.7
 3rd Qu.: 8.936   3rd Qu.:1.656   3rd Qu.:1163.0
 Max.   :11.400   Max.   :1.880   Max.   :1518.0

    azimuth
     0:18
```

The summary includes the call to `variogram`, calculated default values for some of the arguments, and descriptive statistics for the returned values.

To plot the omnidirectional variogram for the coal ash data, shown in figure 4.2, use the generic S-PLUS `plot` function as follows:

```
> trellis.device()
> plot(coal.var1)
```

The `plot` function uses the plotting method `plot.variogram` for objects of class `"variogram"`. For single variograms, a standard plot of gamma versus distance is produced. The axes are set to include the point (0,0). Optionally, the user can set the `xlim` or `ylim` arguments.

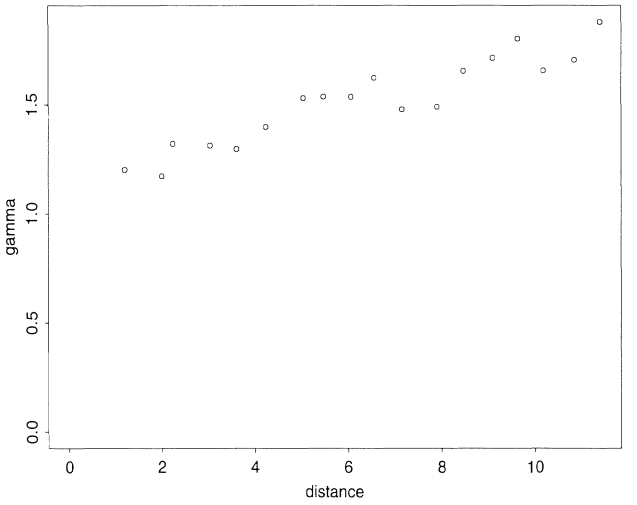

FIGURE 4.2. Omnidirectional empirical variogram for coal ash data.

The omnidirectional variogram for the coal ash data is generally increasing. This may indicate the presence of a large-scale trend or a nonstationary underlying stochastic process. Recall that the EDA of the coal ash data in section 3.2.1 showed an apparent trend in the east-west direction. The coal ash data set requires further analysis before settling on a variogram model.

S+SpatialStats also provides the `covariogram` and `correlogram` functions . The covariogram is defined as:

$$cov(Z(i+h), Z(i)) = C(h), \quad \text{for all } i, i+h \in D$$

The correlogram, $\rho(h)$, is a ratio of covariances and is calculated as

$$\rho(h) = \frac{C(h)}{C(0)} = 1 - \frac{\gamma(h)}{C(0)}$$

where $C(h)$ is the covariance for pairs of points separated by Euclidean distances h (the covariogram), $C(0)$ is the finite variance of the random field, and $\gamma(h)$ is the corresponding variogram. These definitions are for the isotropic case, where h is a scalar; they can be extended to include the case where $\mathbf{h}$ is a vector with both magnitude and direction.

Create and plot the empirical covariogram and correlogram for the coal ash data, shown in figure 4.3, as follows:

```
# set up a 1x2 plotting window
> par(mfrow=c(1,2))
> coal.cov1 <- covariogram(coal ~ loc(x,y), data=coal.ash)
> plot(coal.cov1)
> coal.cor1 <- correlogram(coal ~ loc(x,y), data=coal.ash)
> plot(coal.cor1)
```

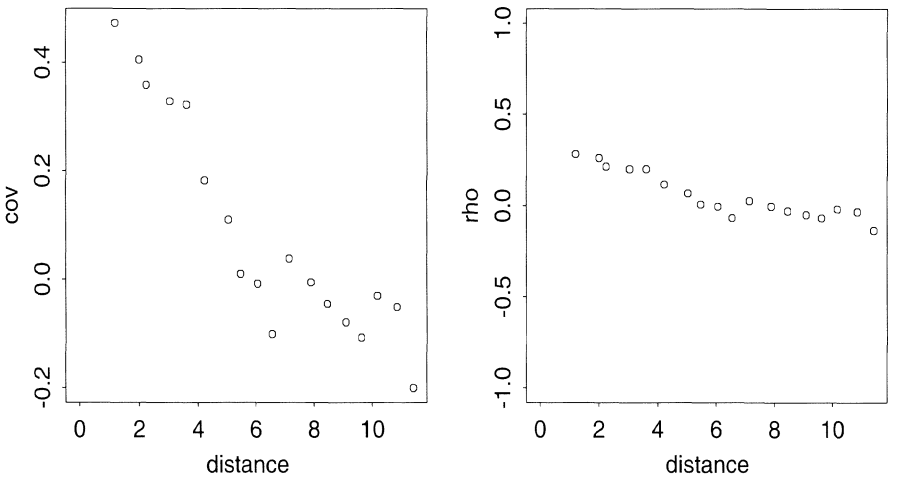

FIGURE 4.3. Omnidirectional empirical covariogram (left) and correlogram (right) for coal ash data.

The `covariogram` function returns an object of class `"covariogram"`, while `correlogram` returns an object of class `"correlogram"`. The objects returned from these functions are similar to that returned from a call to `variogram`. The distance, number of pairs (np), and azimuth are the same quantities. Either a $cov(h)$ or $\rho(h)$ (rho) value is returned, for each distance bin. As for a `"variogram"` object, a call to `plot` will invoke either the `plot.covariogram` or `plot.correlogram` methods. The lower limit of the y-axis of a covariogram defaults to the smaller of 0 and $min(cov)$. The y-axis limits for a correlogram default to $(-1, 1)$. The lower limit of the x-axis for both functions defaults to 0.

Directional Variograms

The `variogram` function can be used to produce *directional variograms*. In this case, $\gamma(h)$ is based on both the magnitude and direction of $\mathbf{h}$. To generate a directional variogram, set the `azimuth` argument to the desired angle (relative to north). Multiple directional variograms can be computed

by specifying a vector of azimuths. Calculate and plot some directional variograms for the coal ash data as follows:

```
> az <- c(0.0, 22.5, 45.0, 67.5, 90.0, 112.5)
> coal.var2 <- variogram(coal ~ loc(x,y), data=coal.ash,
+      azimuth=az, tol.azimuth=11.25)
> plot(coal.var2)
```

For multiple variograms, the method `plot.variogram` produces a multi-panel display that is drawn using the Trellis graphics `xyplot` function. Each panel contains a plot of gamma versus distance for a particular level of azimuth. The axes are set to include the point (0,0). The plots are best displayed on a device started with `trellis.device`.

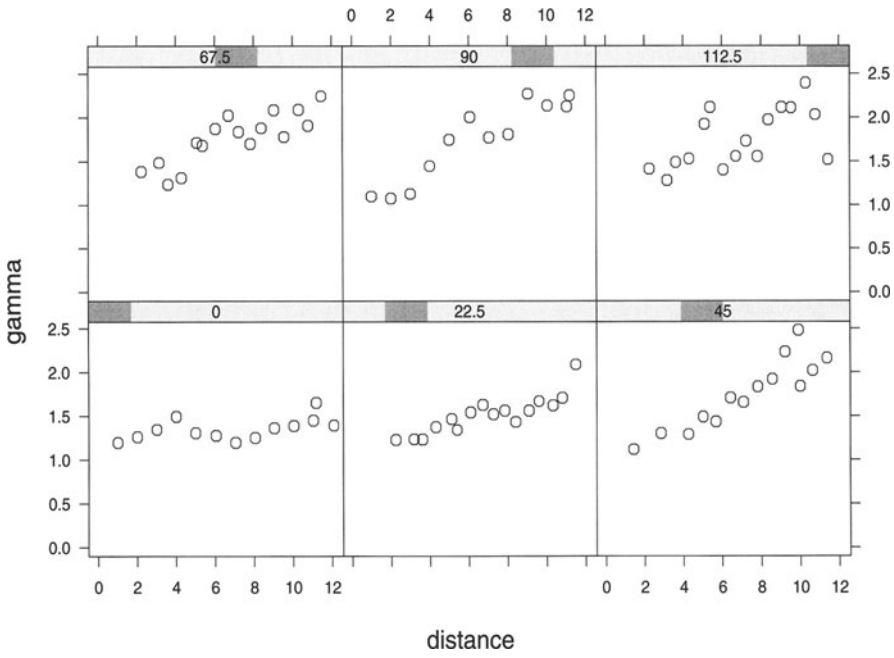

FIGURE 4.4. Directional empirical variograms for coal ash data.

The resulting directional variograms, plotted by `azimuth`, are shown in figure 4.4. The `tol.azimuth` argument is set to 11.25 so each directional variogram is based on all pairs of points that fall within the specified azimuth ± 11.25 degrees.

The plots in figure 4.4 suggest that the north-south (`az=0`) and east-west (`az=90`) directional variograms are different. The north-south direction is

basically flat, indicating little or no autocorrelation. The rest of the directions yield generally increasing variograms, which could be caused by the presence of trend and/or anisotropy, or some other form of nonstationary. See sections 4.1.3 and 4.1.4 for information on identifying and correcting for trend and anisotropy.

Robust Variogram Estimation

The variograms shown thus far have been generated using the classical formulation given by Matheron (1963). The `variogram` function has an option for calculating a robust estimator of the variogram developed by Cressie and Hawkins (1980). The robust estimation is based on the fourth power of the square root of absolute differences as follows:

$$\bar{\gamma}(h) = \frac{\{\frac{1}{2|N(h)|}\sum_{N(h)}|z_i - z_j|^{1/2}\}^4}{0.457 + 0.494/|N(h)|}$$

where $N(h)$ is the set of all pairwise Euclidean distances $i - j = h$, $|N(h)|$ is the number of distinct pairs in $N(h)$, and z_i and z_j are data values at spatial locations i and j, respectively.

The advantage of the robust estimator is that the effect of outliers is reduced, without removing specific data points from a data set. To invoke the robust estimation, set the `method` argument to `"robust"`:

```
> coal.var3 <- variogram(coal ~ loc(x,y), data=coal.ash,
+      azimuth=az, tol.azimuth=11.25, method="robust")
> plot(coal.var3)
```

The directional variograms calculated using the robust estimator are shown in figure 4.5. The robust estimator appears to have smoothed out some of the variability that is present in the variograms (figure 4.4) based on the classical estimator.

4.1.2 Variogram Clouds

The *variogram cloud* is a diagnostic tool that can be used in conjunction with boxplots to look for potential outliers or trends, and to assess variability with increasing distance. Anomalies and nonhomogeneous areas can be detected by looking at short distances that yield high dissimilarities (large values of γ).

A variogram cloud is the distribution of the variance between all pairs of points at all possible distances h. The S+SpatialStats `variogram.cloud` function allows you to specify a variance function of interest; most common

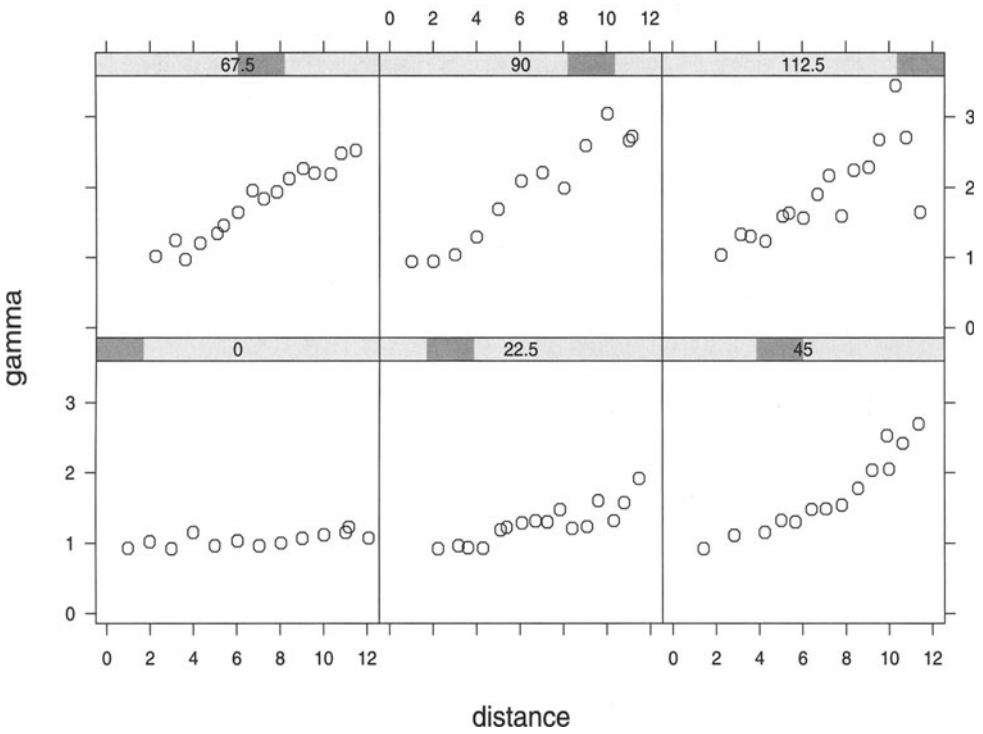

FIGURE 4.5. Robust directional empirical variograms for coal ash data.

are the squared-differences and the square-root-differences. The squared-differences cloud (the default) produces the distribution on which the classical variogram estimator is based, and results in a plot of $(Z_{i+h} - Z_i)^2/2$ versus h. The square-root-differences cloud yields a plot of $\sqrt{(|Z_{i+h} - Z_i|)}/2$ versus h.

Create and plot the default omnidirectional variogram cloud for the scallops data as follows:

```
> scallops.vcloud1 <- variogram.cloud(log(tcatch+1)
+      ~ loc(lat,long), data=scallops)
> # restore full screen plot
> par(mfrow=c(1,1))


> # restore maximum plotting area
> par(pty="m")
> plot(scallops.vcloud1)
```

The resulting squared-differences variogram cloud is shown in figure 4.6. The variogram cloud is extremely dense and shows the existence of many similar data values across all distances. The variability at small distances appears a bit less than that for larger distances.

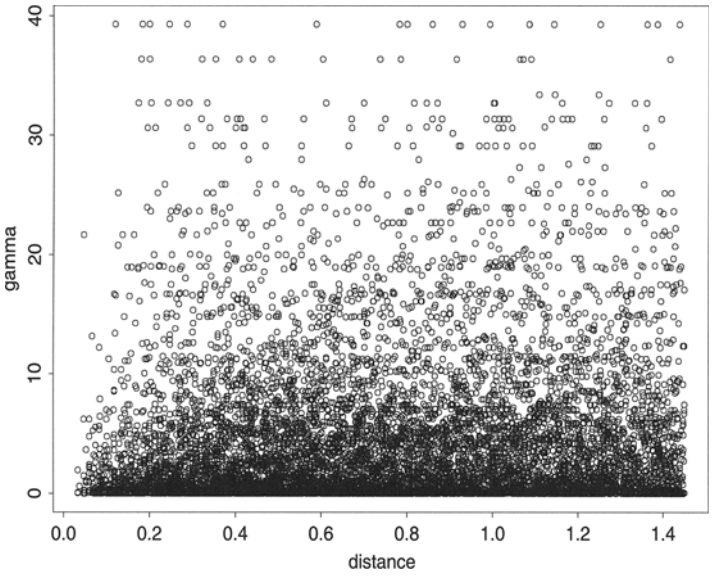

FIGURE 4.6. Squared-differences variogram cloud for scallops data.

An extremely dense variogram cloud may be difficult to interpret. We can reduce the density of the plot by reducing the maximum distance over which the variances are calculated. The optional argument `maxdist` defaults to the distance of reliability (1.5 for the scallops data). Reduce `maxdist` until a visible separation occurs as follows:

```
> par(mfrow=c(2,2))
> scallops.vcloud3 <- variogram.cloud(log(tcatch+1)
+      ~ loc(lat,long), data=scallops, maxdist=.5)
> plot(scallops.vcloud3)
> scallops.vcloud4 <- update(scallops.vcloud3,maxdist=.25)
> plot(scallops.vcloud4)
> scallops.vcloud5 <- update(scallops.vcloud3,
+      maxdist=.125)
> plot(scallops.vcloud5)
> scallops.vcloud6 <- update(scallops.vcloud3,
+     maxdist=.0625)
> plot(scallops.vcloud6)
```

The `update` function is used to create the successive variogram clouds displayed in figure 4.7; `update` adjusts the original call to `variogram.cloud` with the new value of `maxdist`. The `update` function can also be used to easily modify calls to `variogram`.

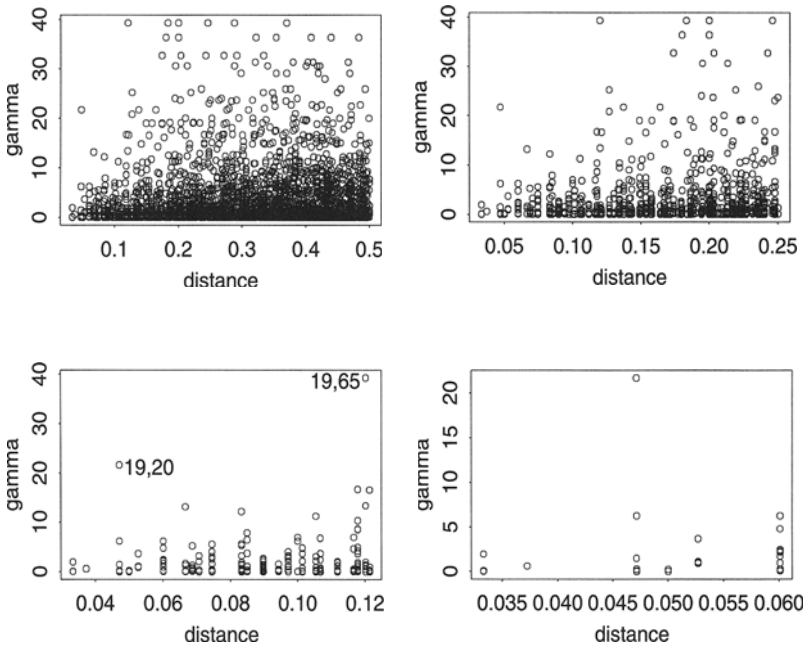

FIGURE 4.7. Variogram clouds for scallops data at various maximum distances.

Location pairs of interest can be interactively identified by using `identify`. The dissimilar (large gamma for the given distance) locations (19,20) and (19,65) identified in figure 4.7 are obtained as follows:

```
> identify(scallops.vcloud5)
```

Once invoked, `identify` allows you to continuously select location pairs by clicking the left mouse button at each point of interest. To exit, click the middle mouse button (right mouse button in Windows) while in the graphics screen. By default, the location pairs are identified on the plot and listed in the S-PLUS command window. The list of points can be saved in an S-PLUS object as follows:

```
> scall.prs <- identify(scallops.vcloud5)
```

Note: In this example, the `identify` function was called before plotting the `scallops.vcloud6` object.

To view a summary while still being able to look for potential outliers, create a boxplot from the variogram cloud as shown in figure 4.8:

```
> boxplot(scallops.vcloud1)
```

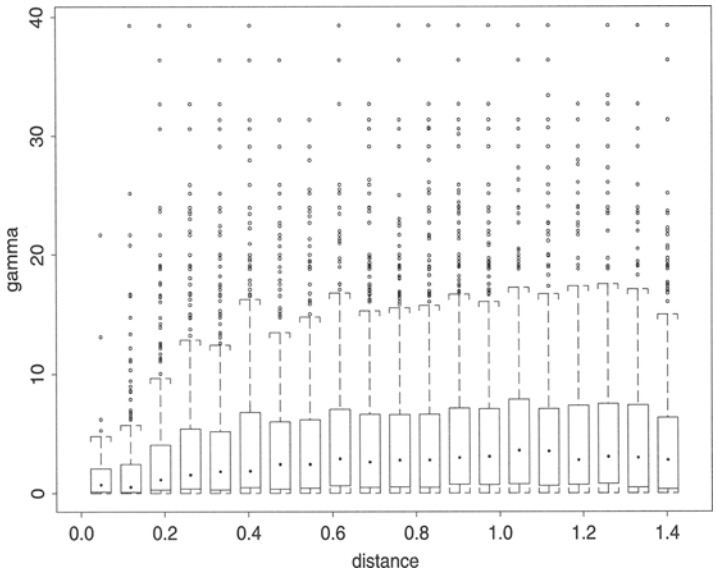

FIGURE 4.8. Boxplots of the squared-difference cloud for scallops data.

By default the data is separated into 20 bins. Many of the points fall outside the top whiskers, indicating potential outliers. The outlying points may be due to the skewed distribution of the variogram cloud as opposed to atypical observations. To check this, create boxplots based on the square-root-differences cloud as follows:

```
> scallops.vcloud2 <- variogram.cloud(log(tcatch+1)
+       ~ loc(lat,long), data=scallops,
+       fun=function(zi,zj) sqrt(abs(zi-zj))/2)
> boxplot(scallops.vcloud2, mean=T, pch.mean="o")
```

The optional `fun` argument is used to calculate the square-root-differences. The `mean` and `pch.mean` arguments are used to include means, in addition to medians, on the boxplots. The boxplots of the square-root-differences cloud are shown in figure 4.9. Only three points are left as potential outliers, all of which occur at distances less than 0.2.

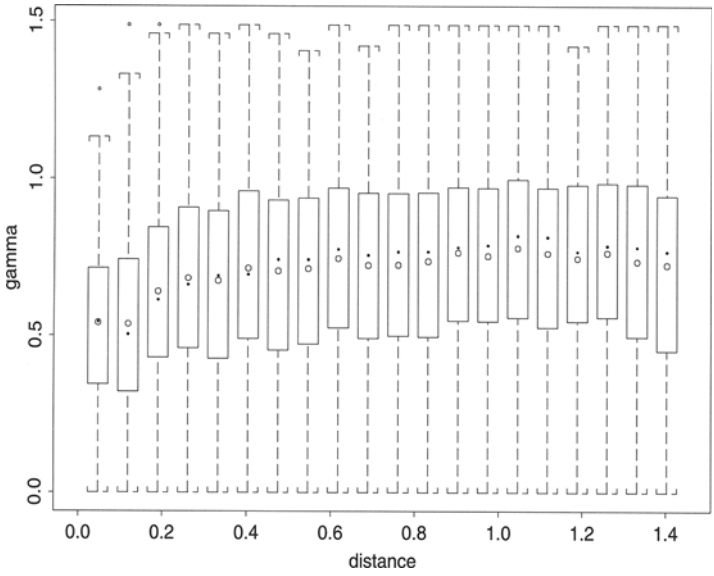

FIGURE 4.9. Boxplots of the square-root-differences cloud for the scallops data.

4.1.3 Detecting and Removing Trends

The existence of the variogram is based on the assumption of *intrinsic stationarity* of the random function, which is defined through first differences as follows:

$$E(Z(i+h) - Z(i)) = 0, \quad \text{for all } i, i+h \in D,$$

and

$$var(Z(i+h) - Z(i)) = 2\gamma(h).$$

Essentially, intrinsic stationarity implies a process with a constant mean and with a variance defined only through the magnitude of h.

Models of geostatistical data, however, are often composed of both large-scale trend or drift, and small-scale random variation. In this case, the random field, $Z(x)$, does not have constant mean, and a variogram based on $Z(x)$ will not meet the necessary assumption.

Detection of possible trends often results from exploratory data analysis. Some graphical methods for elucidating possible trends are shown in sections 3.2.1 and 3.2.2. In section 4.1.1 of this chapter, the use of directional variograms is shown as a way to detect possible trend. Regardless of how a trend is detected, the goal is to appropriately model and detrend the data, and then estimate the variogram for the underlying random process.

Median Polishing

Median polishing is a resistant method for detrending gridded data and is based on an additive decomposition, where

$$\text{data} = \textit{grand} + \textit{row} + \textit{column} + \textit{residual}.$$

Since median polishing assumes an additive trend, the method is not appropriate for trend models with interaction effects between rows and columns. The algorithm iterates successive sweeps of medians out of rows, then out of columns, accumulating them in *row*, *column*, and *grand* effects. The *residuals* are what is left after the algorithm converges.

Median polishing requires data aligned in rows and columns, and thus is naturally suited for use with gridded data; median polishing can be used on non-gridded data only when coerced to a grid. Use the function `twoway` to perform a median polish on the coal ash data as follows:

```
> coal.mp <- twoway(coal~x+y,data=coal.ash)
```

By default, `twoway` will sweep the rows and columns of the coal ash data using medians. Other variations of row and column sweeping, including mean polishing, can be done by setting the optional `trim` argument.

Note: We are using the new generic `twoway` function included with S+SpatialStats. The `twoway.formula` function gets called here.

The call to `twoway` returns the grand median and vectors of row effects, column effects, and residuals. The residuals, in conjunction with the original data, can be used to look for trend as follows:

1. Subtract the median polish residuals from the original data to capture the signal.

   ```
   > coal.signal <- coal.ash$coal-coal.mp$residuals
   ```

2. Convert the vectors for the original data and signal to matrices for use with the `image` function.

   ```
   > coal.mat <- tapply(coal.ash$coal,list(
   +      factor(coal.ash$x),factor(coal.ash$y)),
   +      function(x)x)
   > coalsig.mat <- tapply(coal.signal,list(
   +      factor(coal.ash$x),factor(coal.ash$y)),
   +      function(x)x)
   ```

3. Set up the plotting device and limits for the z values.

```
> motif()
> par(mfrow=c(1,2))
> # force a square plotting region
> par(pty="s")
> zmin <- min(coal.mat[!is.na(coal.mat)],
+      coalsig.mat[!is.na(coalsig.mat)])
> zmax <- max(coal.mat[!is.na(coal.mat)],
+      coalsig.mat[!is.na(coalsig.mat)])
```

4. Produce a grayscale plot of the original data and the signal using a common color scale.

```
> image(coal.mat, zlim=c(zmin,zmax))
> image(coalsig.mat, zlim=c(zmin,zmax))
```

The resulting plots are shown in figure 4.10. An apparent trend in the east-west direction can be seen in the `image` plot of the signal (right-hand plot).

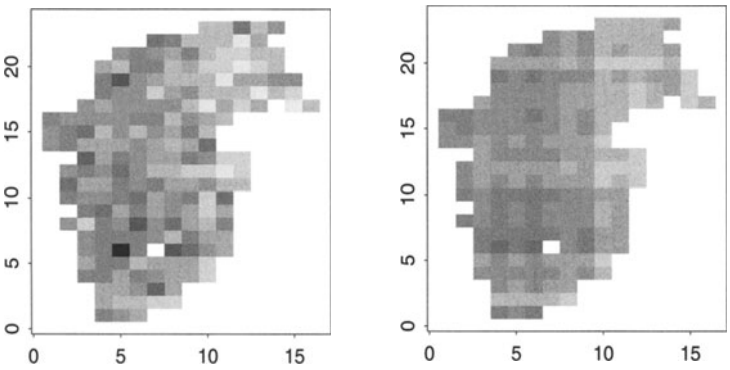

FIGURE 4.10. Results of median polishing of the coal ash data, showing image plots of the original data (left) and the apparent E-W trend in the signal (right).

The effect of the median polish trend removal can be seen by looking at a variogram of the residuals in the east-west direction and comparing it with the corresponding variogram from the original data as follows:

1. Create east-west variograms based on the median polish residuals and the original data, using a common y-axis scale.

```
> coal.var5 <- variogram(coal.mp$residuals
+      ~ loc(x,y),data=coal.ash,
+      azimuth=90, tol.azimuth=11.25)
```

```
> coal.var6 <- variogram(coal~loc(x,y),data=coal.ash,
+       azimuth=90, tol.azimuth=11.25)
```

2. Plot the east-west variograms.

```
> par(mfrow=c(2,1))
> ymax <- max(coal.var5$gamma,coal.var6$gamma)
> plot(coal.var5,ylim=c(0,ymax))
> plot(coal.var6,ylim=c(0,ymax))
```

The east-west variograms based on the median polish residuals and the original data are shown in figure 4.11. Based on the variograms, most of the correlation apparent in the east-west direction seems to be due to the trend.

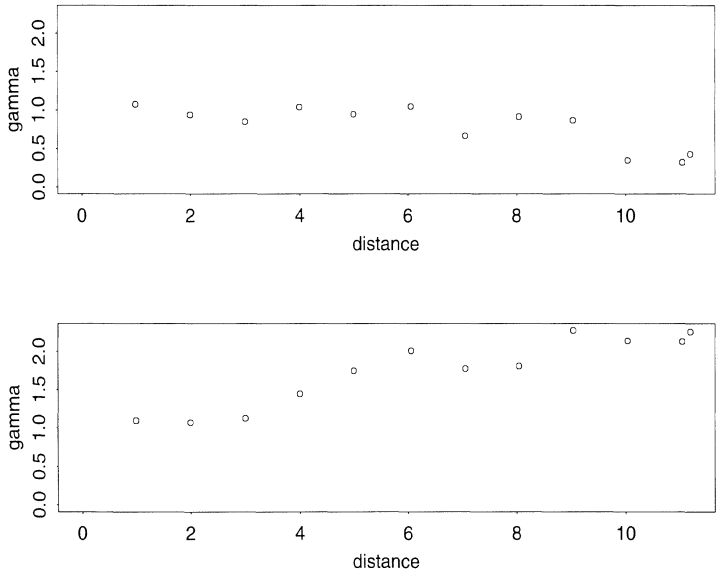

FIGURE 4.11. East-west variogram of the coal ash data calculated from median polish residuals (top) and original data (bottom).

Other Methods

In section 3.2.2, a generalized additive model was fitted to the scallop catch values using smooth functions of latitude and longitude as predictors. Plots of the resulting smooth functions were used to visualize possible trends. Given the observation of trends, a spatial loess model was fitted to the

data using the latitude and longitude as predictors. Residuals from the loess model can be evaluated in a variogram analysis.

Ordinary and generalized linear models are additional methods that may be used to model and remove trend from data.

4.1.4 Anisotropy

Anisotropy is present when the spatial autocorrelation of a process changes with direction; the underlying physical process evolves differently in space. Unlike a variogram from an *isotropic* process, the variogram from an anisotropic process is not purely a function of the distance h, but is a function of both the magnitude and direction of $\mathbf{h}$.

There are two types of anisotropy: *geometric anisotropy* occurs when the range of the variogram changes in different directions, while the sill remains constant; *zonal anisotropy* exists when the sill of the variogram changes with direction.

Identifying and correcting for anisotropy are important because the theoretical variograms used for kriging are based on isotropic models. Geometric anisotropy is generally corrected by a linear transformation of the spatial locations to an equivalent isotropic model. Zonal anisotropy may be corrected by appropriately modeling and detrending the data, or by choosing a *nested* variogram model. One component of a nested model would be an isotropic model fitted to the direction of the zonal anisotropy; the other component might be a geometrically anisotropic variogram function.

Identifying Anisotropy

Directional variograms created with `variogram` can be used to detect anisotropy. Continuing with the rotated scallops data first introduced in section 3.2.2, figure 4.12 shows some directional variograms created as follows:

```
> scallops.dvar1 <- variogram(lgcatch ~ loc(newx,newy),
+      data=scall.rot, azimuth=c(0,45,90,135),
+      tol.azimuth=11.25)
> plot(scallops.dvar1)
```

While the 45 and 135 degree directions yield similar variograms, there are apparent anisotropies in the 0 and 90 degree directions. In the rotated space, the 0 and 90 degree directions correspond roughly to a perpendicular and a parallel coastline orientation, respectively.

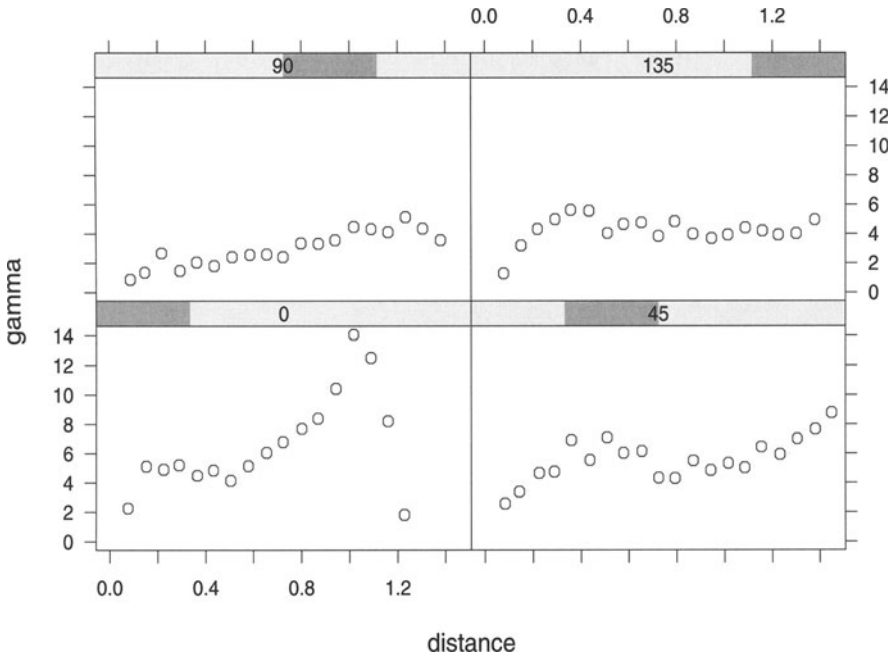

FIGURE 4.12. Directional variograms for rotated scallops data showing anisotropies in the 0 and 90 degree directions.

Disregarding the last four points (due to the low number of pairs) of the 0 degree directional variograms, yields an increasing variogram with no bounds. This could be caused by the apparent trend that was identified through exploratory data analysis using generalized additive models (section 3.2.2).

The 90 degree variogram seems to represent a case of geometric anisotropy. It has a sill of ≈ 5, which looks similar to the sills of the 45 and 135 degree variograms; while its apparent range (≈ 1) is almost twice that of the 45 and 135 degree variograms. Evidently, the autocorrelation parallel to the coastline has a greater range than that for the other directions. This may be expected, given similar environmental conditions parallel to the coastline as opposed to perpendicular to it.

Correcting for Anisotropy

If the anisotropy in the 0 degree variogram in figure 4.12 is the result of trend, then the directional variogram based on detrended data should indicate this. Figure 4.13 shows the same directional variograms, created

using the residuals from a spatial loess model of the rotated scallops data (see section 3.2.2 for a description of the loess model).

```
> scall.res <- scall.rot$lgcatch - predict(loess.scp)
> scallops.dvar2 <- variogram(scall.res
+       ~ loc(scall.rot$newx, scall.rot$newy),
+       azimuth=c(0,45,90,135), tol.azimuth=11.25,
+       method="robust")
> plot(scallops.dvar2)
```

The 0 degree variogram is no longer increasing and appears to have a range and sill similar to the 45 degree variogram. The 90 degree variogram still has a range considerably greater than the variograms in other directions. The 135 degree variogram may be a pure nugget effect, showing little or no spatial correlation. The sills of all the variograms are approximately the same and are reduced from the original directional variograms. This is consistent with the removal of trend.

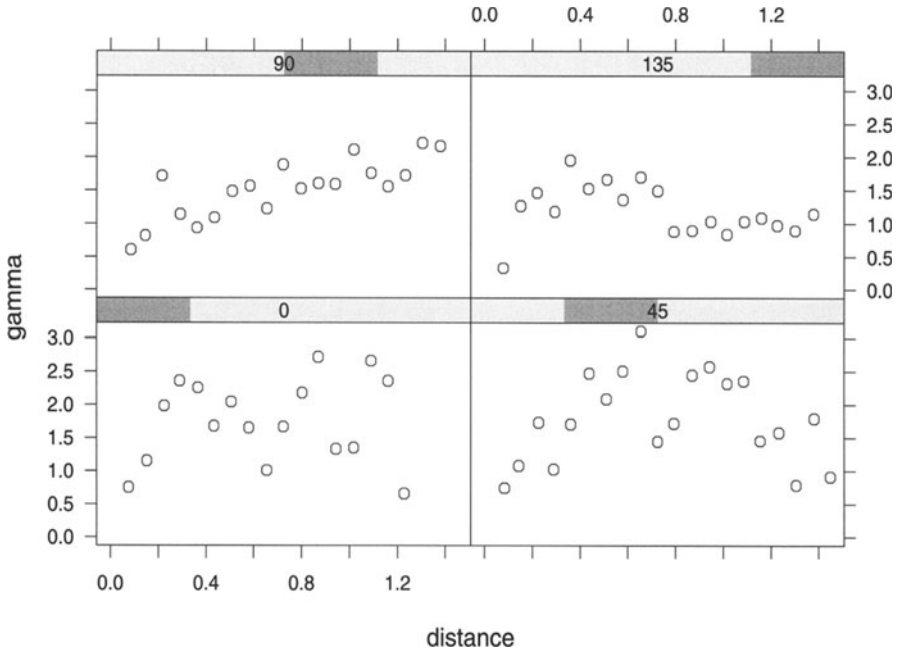

FIGURE 4.13. Directional variograms for residuals from a loess model on the rotated scallops data.

To better define the geometric anisotropy, a description of how the range changes in various directions is needed. This can be evaluated by looking

at contours of gamma for various distances and directions. One way to do this is as follows:

1. Write a panel function that calculates the distance value for a specified value of gamma. Use a loess smooth to approximate the variogram function and then the `approx` function to determine the distance corresponding to a given value of gamma on the loess curve.

```
> panel.gamma0 <-
+ function(x, y, gamma0 = gamma0, span = 2/3, ...)
+ {
+    lofit <- loess.smooth(x, y, span = span)
+    panel.xyplot(x, y, ...)
+    panel.xyplot(lofit$x, lofit$y, type = "l")
+    dist0 <- approx(lofit$y, lofit$x,
+       xout = gamma0)$y
+    segments(0, gamma0, dist0, gamma0)
+    segments(dist0, 0, dist0, gamma0)
+    parusr <- par()$usr
+    text(parusr[2] - 0.05 * diff(parusr[1:2]),
+       parusr[3] + 0.05 * diff(parusr[3:4]),
+       paste("d0=", format(round(dist0, 4)),
+       sep = ""), adj = 1)
+ }
```

   For convenience, this panel function is included with S+SPATIALSTATS; however, a help file is not provided.

2. Subset `scallops.dvar2` to include only points for distances less than 0.9 in the 0 and 45 degree directions, due to the low numbers of pairs.

```
> scallops.aniso <- scallops.dvar2[
+      (scallops.dvar2$distance < 0.9 &
+      scallops.dvar2$azimuth == 0) |
+      (scallops.dvar2$distance < 0.9 &
+      scallops.dvar2$azimuth == 45) |
+      scallops.dvar2$azimuth == 90 |
+      scallops.dvar2$azimuth == 135,]
```

3. Plot the variograms with interpolated distances for gamma = 1.4, using `xyplot` with the panel function `panel.gamma0`.

```
> xyplot(gamma ~ distance | azimuth,
+      data=scallops.aniso, panel=panel.gamma0,
+      gamma0=1.4)
```

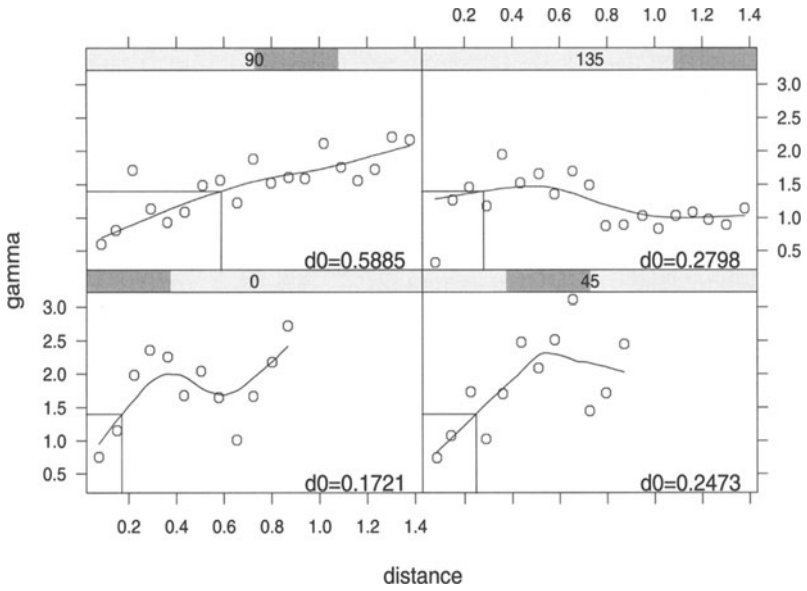

FIGURE 4.14. Directional variograms for scallops loess residuals with interpolated distances for $\gamma(1.4)$.

Figure 4.14 shows the interpolated distances for $\gamma(1.4)$.

The interpolated distances, h, are similar for the 45 and 135 degree directions. The most prominent difference is in the 90 degree direction.

Note: The independence assumption for the loess smoother is not met for the variogram values. It is only used to get a first approximation, which seems sufficient to gauge the anisotropy.

The interpolated distances can be used to construct a rose diagram (Isaaks and Srivastava, 1989), providing another technique for visualizing and defining the anisotropy. Rose diagrams are equivalent to showing the distance in several directions along a given gamma contour. Create the rose diagram shown in figure 4.15 as follows:

1. Scale the interpolated distances relative to the largest value of h. In this case, the scaled distances are .292, .420, 1.000, and .475 for 0 through 135 degrees, respectively:

```
> dscale <- .5885
> d0 <- .1721/dscale
> d45 <- .2473/dscale
> d90 <- 1
> d135 <- .2798/dscale
```

2. Determine the points on the unit xy surface that correspond to the scaled distances in each direction. For the 0 and 90 degree directions, the points are (0,.292) and (1,0). For the 45 degree direction, the point can be calculated from the equation for a circle, $x^2 + y^2 = r^2$, where r is the radius, or the distance in this case. For an azimuth of 45 degrees, $x = y$, so the calculation simplifies to $x^2 + x^2 = r^2$. The point for the 45 degree direction is (.297,.297). The point for the 135 degree direction is (.336,.336).

3. Use the functions `plot` and `segments` to generate the rose diagram:

```
> plot(0,0,type='n', axes=F, xlim=c(-1,1),
+      ylim=c(-1,1),
+      xlab="along scall.rot$newx",
+      ylab="along scall.rot$newy")
> segments(0,0,1,0)
> segments(0,0,0,.292)
> segments(0,0,.297,.297)
> segments(0,0,.336,-.336)
> segments(0,0,-1,0)
> segments(0,0,0,-.292)
> segments(0,0,-.297,-.297)
> segments(0,0,-.336,.336)
```

Rose diagrams are symmetric (since variograms are symmetric); the segments are extended in corresponding 180 degree directions to complete the diagram.

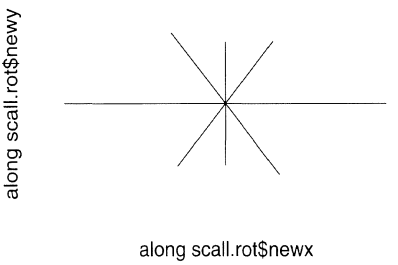

FIGURE 4.15. Rose diagram for scallops loess model residuals based on interpolated distances for $\gamma(1.4)$.

The shape of the rose diagram is approximately elliptical with the major axis in the 90 degree direction. This further defines the form of the geometric anisotropy, which can be corrected by a linear transformation.

S+SpatialStats provides the function `anisotropy.plot` to perform linear transformations of the data and to generate the resulting isotropic variograms. Correct for the geometric anisotropy in the scallops data as follows:

```
> anisotropy.plot(scall.res ~ loc(scall.rot$newx,
+       scall.rot$newy), angle=90,
+       ratio=c(2.25,2.5,2.75,3.0,3.25,3.5),
+       method="robust", layout=c(2,3))
```

The optional argument `angle` is the clockwise angle of rotation, in degrees, by which the y axis must be rotated to become parallel to the major axis of the ellipse. The optional argument `ratio` is the ratio of the length of the major to minor axes of the ellipse. For the scallops data, the axes must be rotated a full 90 degrees, so we set `angle` to 90. Based on a gamma contour of 1.4, the ratio of the major to minor ellipse axes is approximately 3. Thus, we set `ratio` to a range of values around 3 for comparison. Figure 4.16 shows the variograms resulting from the linear transformations.

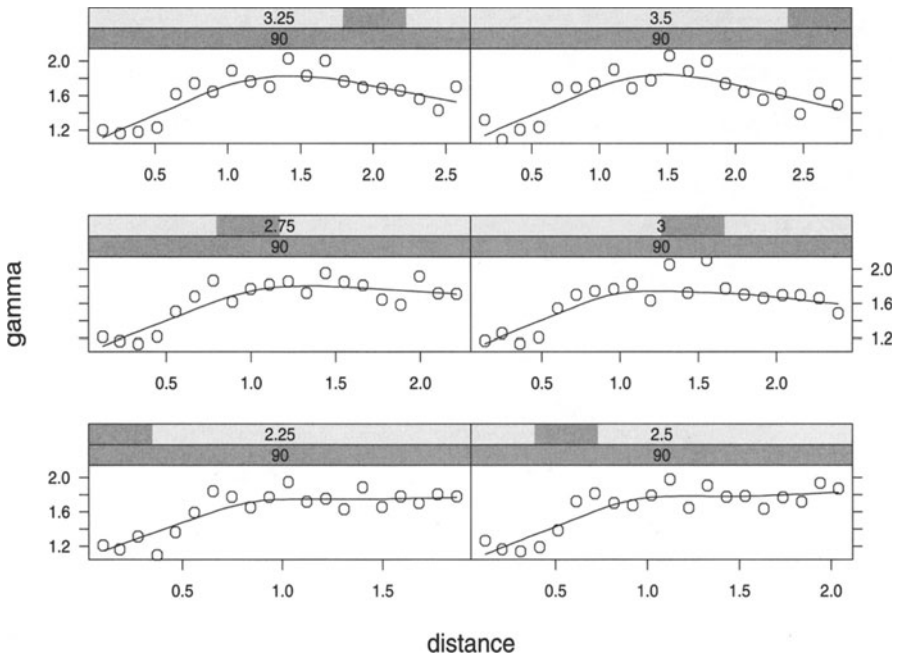

FIGURE 4.16. Scallops variograms corrected for trend and geometric anisotropy.

The resulting variograms, corrected for geometric ansiotropy and trend, are similar. The specific choice as to which variogram to use for model fitting is up to the researcher.

4.2 Modeling the Empirical Variogram

In section 4.3 we show how the variogram is incorporated into kriging equations for use in making predictions and calculating kriging prediction variances. In order to ensure that the variance of predicted values is positive, the empirical variogram must be replaced with a theoretical variogram function.

4.2.1 Theoretical Variogram Models

S+SpatialStats provides functions for the common theoretical variogram models. The *exponential*, *spherical*, and *gaussian* models are bounded variogram functions. The *linear* and *power* models increase without bounds. Plot examples of the various theoretical models shown in figure 4.17 as follows:

```
> par(mfrow=c(3,2))
> vdist <- 1:15
> vrange <- 7
> plot(vdist,exp.vgram(distance=vdist, range=vrange),
+      type="l")
> plot(vdist,spher.vgram(distance=vdist, range=vrange),
+      type="l")
> plot(vdist,gauss.vgram(distance=vdist, range=vrange),
+      type="l")
> plot(vdist,linear.vgram(distance=vdist, slope=.3),
+      type="l")
> plot(vdist,power.vgram(distance=vdist, slope=.3,
+      range=.5), type="l")
```

All of the theoretical variogram functions require specification of a `distance` vector. The exponential, spherical and gaussian models also require a `range` value. The linear and power models require a `slope` value. The `nugget` argument is optional for all models and defaults to 0. In addition, there is an optional `sill` argument for the bounded functions (default = 1) and a `range` argument required for the power function.

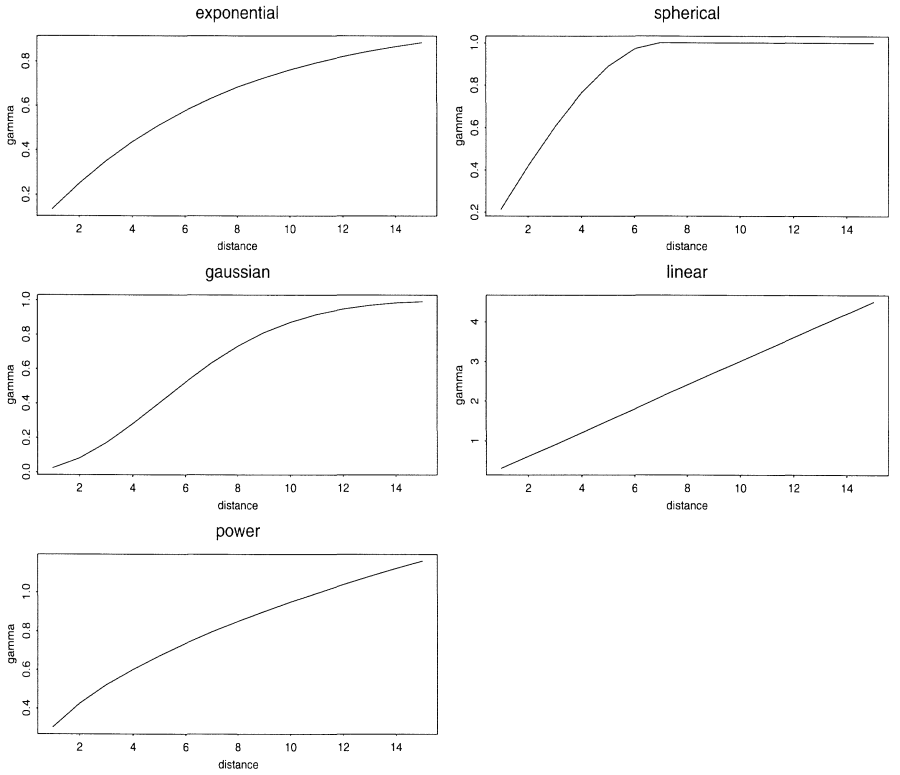

FIGURE 4.17. Theoretical variogram models in S+SPATIALSTATS.

For `exp.vgram` the range is defined as approximately one-third the *apparent* range, where the apparent range is the observed distance h above which data no longer appear to be correlated. For the spherical and gaussian models, the `range` argument is the apparent range. For the power model, the `range` argument is the exponent of the distance. The `sill` argument represents the *absolute* sill; the absolute sill is estimated as the sill of the empirical variogram minus any nugget effect.

4.2.2 Fitting a Theoretical Variogram Model

Fitting a theoretical variogram model to an empirical variogram is often done by eye. Even if a numerical fitting routine is to be used, looking at an initial fit by eye can be useful. The initial model type is chosen based on the shape of the empirical variogram and the researcher's belief as to the nature of the operating processes. Initial values for the range, sill, and nugget effect parameters are also chosen from the empirical var-

iogram. The fit of the theoretical model can be viewed and interactively updated using the S+SPATIALSTATS function `model.variogram`. Within a call to `model.variogram`, parameter values can be modified, and the results overlaid onto the empirical variogram until a satisfactory fit is obtained.

Fit a theoretical variogram model to the scallops data variogram that was corrected for geometric anisotropy (section 4.1.4) as follows:

1. Calculate the empirical variogram based on the appropriate correction (transformation) for geometric anisotropy. Based on the plots in figure 4.16, a correction using an angle of 90 degrees with a ratio of 2.25 yields a reasonable variogram. Create the transformed variogram as follows:

```
> scallops.finalvar <- variogram(scall.res
+      ~loc(scall.rot$newx, scall.rot$newy,
+      angle=90, ratio=2.25), method="robust")
```

2. Choose a theoretical model and initial parameter values. Based on the general shape of the empirical variogram (figure 4.16, bottom left), we will start with a spherical model having range = .8, sill = 1.75, and nugget effect = .5.

3. Call `model.variogram` with the initial model choice.

```
> model.variogram(scallops.finalvar,fun=spher.vgram,
+           range=.8, sill=1.75-.5, nugget=.5)
Select a number to change a parameter (or 0 to exit):
 Current objective = 0.4814
1: range - current value: 0.8
2: sill - current value: 1.25
3: nugget - current value: 0.5
Selection:
```

   The `model.variogram` function displays the initial theoretical and empirical variograms (figure 4.18, top left) and an interactive menu that allows changes to be made in the parameter values for the current model type. Choose options 1, 2, or 3 to modify the range, sill, or nugget parameters for the spherical variogram. After each change, the updated theoretical model is automatically plotted. The `model.variogram` function also includes a measure of model fit through the optional argument `objective.fun`. By default, the objective function is the residual sum of squares between the theoretical model and the empirical variogram. For each iteration of parameter changes, a new objective value is calculated and displayed on both

the command screen and the plot. The user may implement a custom objective function. Enter `0` at the `Selection:` prompt to exit `model.variogram`.

4. Adjust the fit, if necessary, by modifying parameter values. For example, it looks as though we may need to increase the nugget effect (and appropriately adjust the sill) and decrease the range. Do this as follows:

```
Selection: 3
New nugget : .7
Select a number to change a parameter (or 0 to exit):
 Current objective = 0.9705
1: range - current value: 0.8
2: sill - current value: 1.15
3: nugget - current value: 0.7
Selection: 2
New sill : 1.05
Select a number to change a parameter (or 0 to exit):
 Current objective = 0.3352
1: range - current value: 0.8
2: sill - current value: 1.05
3: nugget - current value: 0.7
Selection: 1
New range : .75
Select a number to change a parameter (or 0 to exit):
 Current objective = 0.3522
1: range - current value: 0.75
2: sill - current value: 1.05
3: nugget - current value: 0.7
Selection: 0
```

The resulting fitted models are shown in figure 4.18. Using the interactive menu, the nugget was first increased to 0.7 (top right); to compensate for the increased nugget, the sill was decreased to 1.05 (bottom left); the range was decreased to 0.75 (bottom right). The residual sum of squares decreased with each iteration of a fitted model, until the range was decreased. The final model of choice is a spherical variogram with range = 0.8, sill = 1.75, and nugget effect = 0.7.

The function `model.variogram` can also be used to interactively fit covariograms and correlograms.

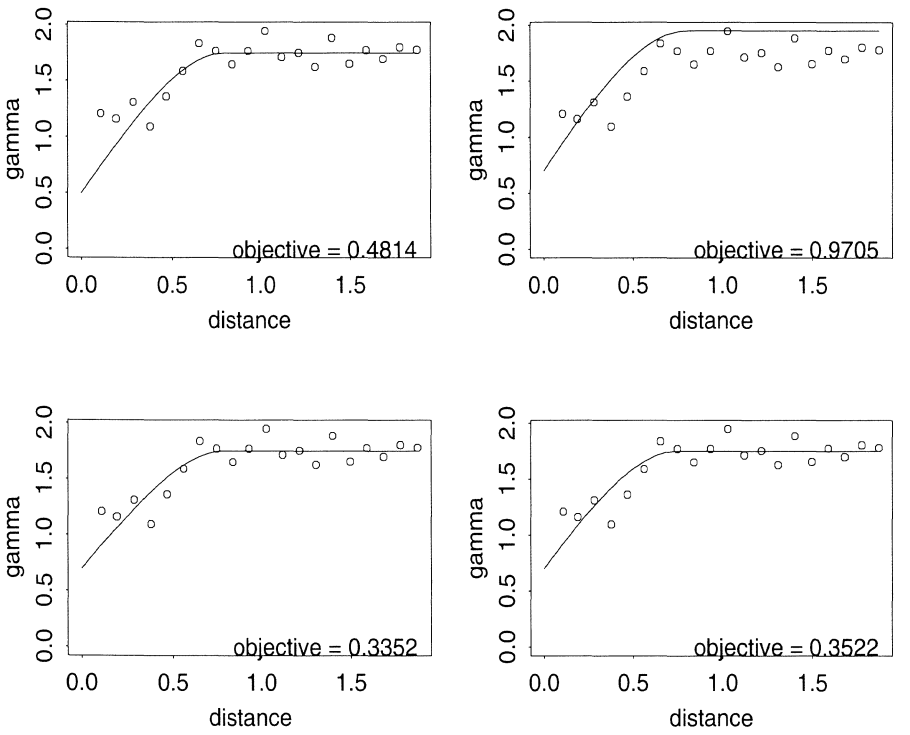

FIGURE 4.18. Scallops empirical variogram with several iterations resulting in a fitted spherical model.

Fitting Using Nonlinear Least Squares

Variogram models can be fit by optimization techniques, typically some form of nonlinear least squares. The usual statistical assumptions for nonlinear regression models are not valid for fitting variograms since the variogram values at different lags are not independent. Cressie (1985) describes weighted least squares and generalized least squares approaches for variogram fitting that try to account for the special structure of the variogram values. Zimmerman and Zimmerman (1991) compare several estimation methods and conclude that ordinary nonlinear least squares or some form of weighted nonlinear least squares is usually as good as many of the more complicated and computationally intensive methods.

The use of nonlinear least squares variogram fitting is demonstrated on the coal ash data. Earlier (section 3.2.1) it was determined that there is a trend in the east-west direction of this data. For illustration purposes, we will fit a spherical variogram model to only the north-south data as follows:

1. Compute and plot the north-south variogram.

```
> coal.varns <- variogram(coal ~ loc(x, y),
+      data = coal.ash, azimuth = 0,
+      tol.azimuth = .01, lag = 1)
> plot(coal.varns)
```

   The resulting variogram is shown in figure 4.19.

2. Write a function defining the residuals for which the sums of squares will be minimized, using the spherical variogram function `spher.vgram`.

```
> spher.fun <- function(gamma, distance, range,
+      sill, nugget)
+      gamma - spher.vgram(distance, range = range,
+      sill = sill, nugget = nugget)
```

3. Choose starting values. From the variogram plot of `coal.varns`, we choose range = 4.0, sill = 0.2, and nugget effect = 0.8.

4. Call `nls` using the function `spher.fun` defined above.

```
> coal.nl1 <- nls( ~ spher.fun(gamma, distance,
+      range, sill, nugget), data = coal.varns,
+      start = list(range = 4, sill = 0.2, nugget=.8))
> coef(coal.nl1)
    range     sill   nugget
 3.443335 0.240059 1.093492
```

5. Add the fitted model to the empirical variogram (figure 4.19).

```
> lines(coal.varns$dist, spher.vgram(coal.varns$dist,
+       range=3.443335,sill=.240059,nugget=1.093492))
```

Cressie (1985) suggests minimizing the weighted sum of squares :

$$\sum_{j=1}^{K} |N(h(j))| \left\{ \frac{\gamma(h(j))}{\gamma(h(j);\boldsymbol{\theta})} - 1 \right\}^2,$$

where $|N(h(j))|$ is the number of distinct pairs in lag j, K is the number of lags in the empirical variogram, $\gamma(h(j))$ is the value of the empirical variogram at lag j, and $\gamma(h(j);\boldsymbol{\theta})$ is the known variogram model with unknown parameters $\boldsymbol{\theta}$.

This weighting function can be easily incorporated into `nls` by defining the residual function to include the weights:

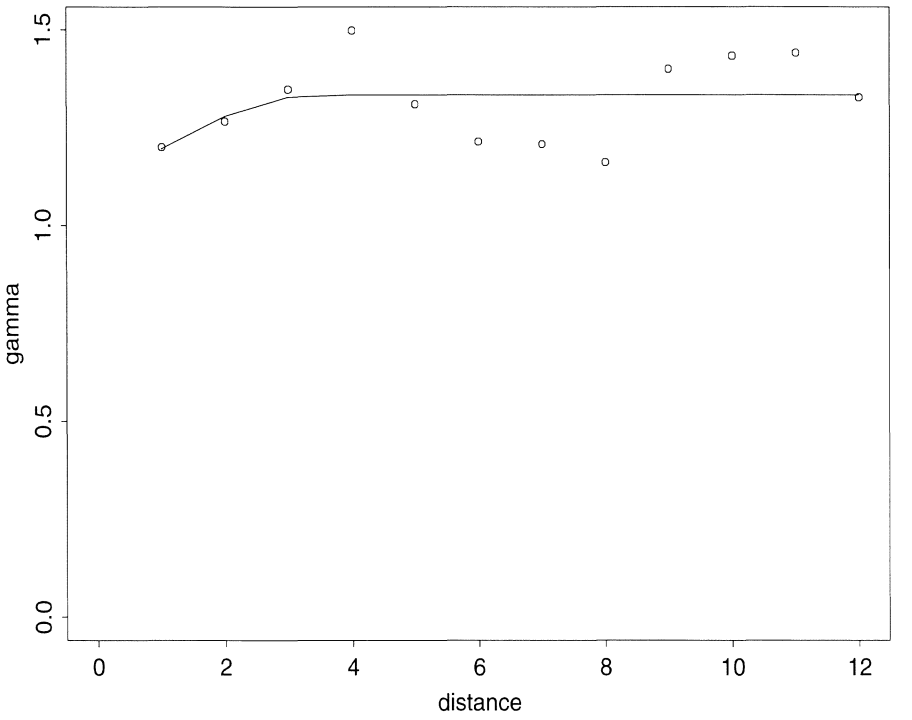

FIGURE 4.19. Coal ash empirical variogram (open circles) with a spherical fitted model (line) based on estimates using nonlinear least squares.

```
> spher.wfun <-
+ function(gamma, distance, np, range, sill, nugget)
+ {
+     gammahat <- spher.vgram(distance, range = range,
+     sill = sill, nugget = nugget)
+          sqrt(np) * (gamma/gammahat - 1)
+ }
```

Calculate new variogram parameter estimates for the coal ash data, incorporating the weights, as follows:

```
> coal.nl2 <- nls( ~ spher.wfun(gamma, distance,
+       np, range, sill, nugget), data = coal.varns,
+       start = list(range = 4, sill = 0.2, nugget=.8))
> coef(coal.nl2)
    range      sill   nugget
 3.658897 0.2427316 1.099064
```

In this case, the coefficients from the weighted nonlinear least squares are not much different than the unweighted coefficients.

Other functions for nonlinear fitting in S-PLUS include `ms` and `nlminb`. See the individual help files for details on these functions. For additional information on fitting nonlinear models in S-PLUS see Bates and Chambers (1992) or Venables and Ripley (1994).

4.3 Kriging

Kriging is a linear interpolation method that allows predictions of unknown values of a random function from observations at known locations. Kriging incorporates a model of the covariance of the random function when calculating predictions of the unknown values. S+SPATIALSTATS provides functions to perform two types of kriging: *ordinary* and *universal*. Ordinary kriging uses a random function model of spatial correlation to calculate a weighted linear combination of available samples, for prediction of a nearby unsampled location. Weights are chosen to ensure that the average error for the model is zero and that the modeled error variance is minimized (Isaaks and Srivastava, 1989). Universal kriging is an adaptation of ordinary kriging that accomodates trend. Universal kriging can be used to both produce local estimates in the presence of trend, and to estimate the underlying trend itself. Universal kriging with a constant mean is equivalent to ordinary kriging.

4.3.1 Ordinary Kriging

Ordinary kriging in 2-dimensions is performed in S+SPATIALSTATS by using the `krige` and `predict.krige` functions: `krige` uses the kriging response variable, spatial locations, and a theoretical covariance function to set up kriging matrices for the predictions; `predict.krige` uses the output from `krige` to compute kriging predictions and standard errors for unsampled locations specified by the user. The theoretical covariance function is based on a theoretical variogram, and is defined for the exponential, spherical and gaussian models. The linear and power variogram models do not have corresponding covariance models since they are unbounded. A linear covariance model can be calculated, however, by subtracting the values of a linear variogram from a large number (the value of gamma corresponding to the greatest distance over which predictions will be made).

Perform ordinary kriging of the scallops data as follows:

```
> scallops.krige <- krige(scall.res~loc(newx,newy,
+      90,2.25), data=scall.rot, covfun=spher.cov,
+      range=.8, sill=(1.75-.7), nugget=.7)
```

The kriging variable is `scall.res`, the residuals from a spatial loess model. The spatial locations are the rotated coordinates `newx` and `newy`. The spatial correlation is modeled as spherical covariance based on the spherical variogram model fitted to the empirical variogram (figure 4.18, bottom left). As with `spher.vgram`, the `sill` argument for `spher.cov` is specified as the sill minus the nugget effect.

The `krige` function returns an object of class `"krige"` that includes a summary of the call and the calculated coefficients:

```
> scallops.krige
Call:
krige(formula = scall.res ~ loc(newx, newy, 90, 2.25),
     data = scall.rot, covfun = spher.cov, range = 0.8,
     sill = (1.75 - 0.7), nugget = 0.7)

Coefficients:
   constant
 -0.1873672

Number of observations: 148
```

Kriging predictions at a set of unsampled spatial locations can now be made using `predict.krige`. There are two ways to define unsampled locations for the prediction. One way involves the `newdata` argument; `newdata` is a data frame or list containing the spatial locations for the predictions. Alternatively, a grid of points can be generated using the `grid` argument; `grid` is a list of two vectors (one for each axis), specifying the minimum, maximum, and number of points. For either `newdata` or `grid`, the names must match the names of the locations used in the call to `krige`. By default, a set of prediction locations are generated on a 30×30 grid defined by the minimum and maximum spatial coordinates of the sampled locations. Calculate kriging predictions for the scallops residuals using the default locations as follows:

```
> scallops.pkrige1 <- predict(scallops.krige)
```

The generic `predict` function calls the method `predict.krige` for an object of class `"krige"`. The call to `predict` returns a data frame with four columns corresponding to the x, y locations of the predictions, the predictions, and the standard errors of the predictions. Plot the sampled and prediction locations as follows:

```
> plot(scall.rot$newx, scall.rot$newy,
+      xlab="newx", ylab="newy", pch=16)
> points(scallops.pkrige1$newx, scallops.pkrige1$newy,
+      pch="+")
```

Figure 4.20 shows the original sampled locations and the default prediction locations returned from `predict`.

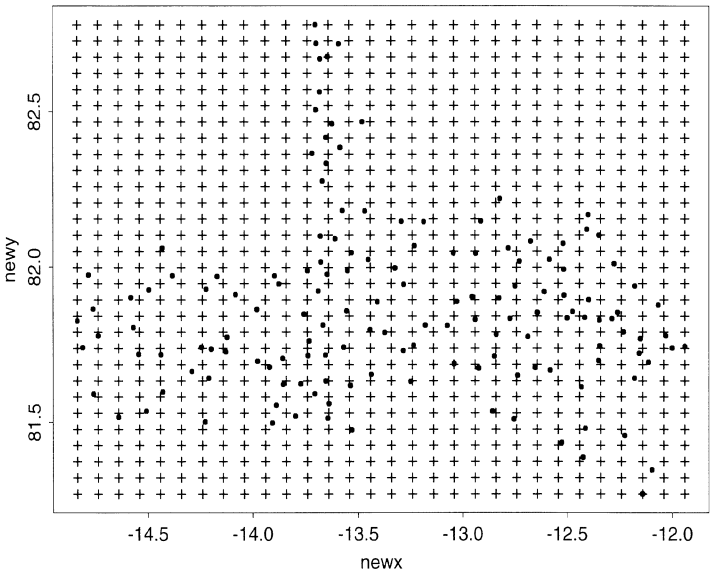

FIGURE 4.20. Original scallop sampling locations (closed circles) and default prediction locations (+'s).

These prediction locations may not be satisfactory for the scallops data. There are too many locations close to shore, far from sampled locations; some of the default locations are on land. In order to use the `grid` argument, minimum and maximum locations corresponding to the main rectangular sampling area should be used. Alternatively, S+SpatialStats and S-Plus have several functions that can be used to generate prediction locations within polygonal boundaries. The function `chull` can be used to calculate the convex hull of a sampling region; `locator` can be used to interactively construct a user-defined polygon around, or within a sampling region. Create the polygonal boundaries shown in figure 4.21 as follows:

```
> par(mfrow=c(1,2))
> par(pty="s")
>
```

```
> plot(scall.rot$newx, scall.rot$newy, pch=16)
> scallops.chull <- chull(scall.rot$newx, scall.rot$newy)

> polygon(scall.rot$newx[scallops.chull],
+       scall.rot$newy[scallops.chull], density=0)
>
> plot(scall.rot$newx, scall.rot$newy, pch=16)
> scallops.poly <- locator(type = "l")
```

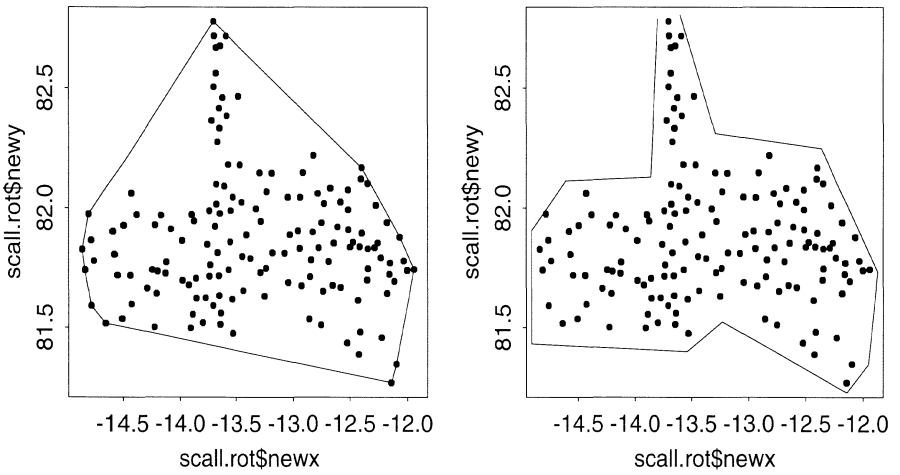

FIGURE 4.21. Convex hull (left) and user-defined polygon (right) based on scallops data sampling locations.

The call to `chull` returns the indices of the location vectors which correspond to the vertices of the convex hull. The `polygon` function adds the convex hull to the current plot. By default, `polygon` shades in the convex hull; setting the optional `density` argument to 0 causes only the outline to be drawn. The `locator` function, with the optional `type` argument set to `"l"`, interactively draws a polygon as user selected points are identified. Select boundaries for the polygon using the *left* mouse button. Exit `locator` from within the graphics window by using the *middle* mouse button in UNIX or the *right* mouse button in Windows. There is no need to close a user-defined polygon.

Prediction locations can be defined within the bounds of the convex hull or user-defined polygon using `poly.grid`. Create the prediction locations shown in figure 4.22 as follows:

```
> par(mfrow=c(1,2))
> par(pty="s")
```

```
>
> plot(scall.rot$newx, scall.rot$newy, pch=16)
> predict.loc1 <- poly.grid(cbind(
+      scall.rot$newx[scallops.chull],
+      scall.rot$newy[scallops.chull]), nx=20, ny=20)
> points(predict.loc1,pch="+")
>
> plot(scall.rot$newx, scall.rot$newy, pch=16)
> predict.loc2 <- (poly.grid(scallops.poly,
+      size=c(0.08,0.05)))
> points(predict.loc2, pch="+")
```

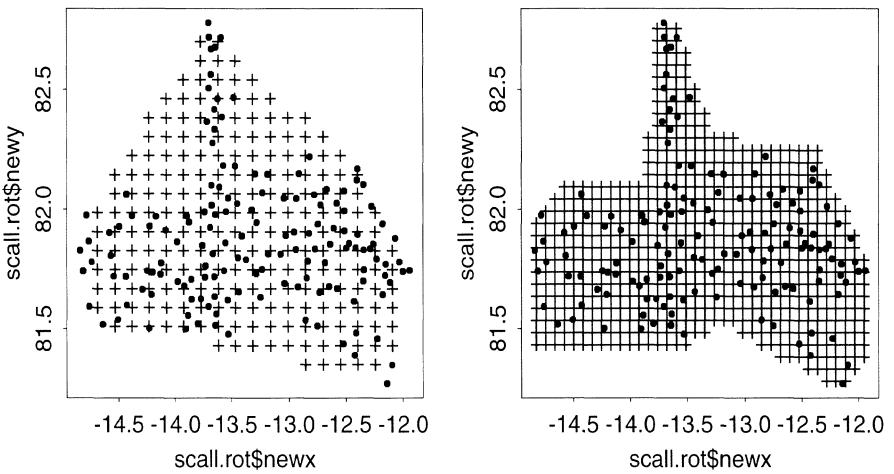

FIGURE 4.22. Prediction locations for the convex hull (+'s, left) and user-defined polygon (+'s, right).

In the left plot (figure 4.22) the prediction locations were specified by setting the `nx` and `ny` arguments of `poly.grid` resulting in a $nx \times ny$ grid. The locations on the right plot were created using the optional `size` argument to create a grid with spacing of 0.08 in the x direction and 0.05 in the y direction.

Perform ordinary kriging of the scallops residuals, using the prediction locations within the user-defined polygon in figure 4.22, as follows:

1. Convert `predict.loc2`, the output from a call to `poly.grid`, to a data frame. Change the names of the prediction locations to match those used in the call to `krige`.

Note: A data frame is not required for the function `predict.krige`, but a data frame is required for `predict.loess` which will be called later.

```
> predict.loc2 <- as.data.frame(predict.loc2)[c(1,2)]
> names(predict.loc2) <- c("newx","newy")
```

2. Calculate predictions.

```
> scallops.pkrige2 <- predict(scallops.krige,
+       newdata=predict.loc2)
```

3. Look at a summary of the predictions.

```
> summary(scallops.pkrige2)
      newx             newy            fit
 Min.   :-14.83   Min.   :81.28   Min.   :-3.07000
 1st Qu.:-13.92   1st Qu.:81.58   1st Qu.:-0.33770
 Median :-13.43   Median :81.84   Median :-0.09671
 Mean   :-13.36   Mean   :81.85   Mean   :-0.22850
 3rd Qu.:-12.69   3rd Qu.:82.04   3rd Qu.: 0.19210
 Max.   :-11.95   Max.   :82.75   Max.   : 0.96700

     se.fit
 Min.   :0.9445
 1st Qu.:1.0030
 Median :1.0320
 Mean   :1.0490
 3rd Qu.:1.0800
 Max.   :1.2540
```

In this case, `newx` and `newy` are the prediction locations, `fit` is the kriging predictions for the residuals, and `se.fit` is the corresponding kriging prediction standard errors. To generate a predicted surface for the scallops catch data, add predictions from the spatial loess model to the kriged residuals as follows:

```
> scall.lo <- predict(loess.scp, predict.loc2)
> scall.pred <- scallops.pkrige2$fit + scall.lo
```

Create contour maps for the predicted surface and relative standard errors as follows:

1. Map the scallop predictions and kriging prediction standard errors to matrices bounded by the locations in `predict.loc2`.

```
> xmat <- sort(unique(predict.loc2$newx))
> ymat <- sort(unique(predict.loc2$newy))
> scall.predmat <- matrix(NA, length(xmat),
+      length(ymat))
> scall.predmat[cbind(match(predict.loc2$newx,
+      xmat), match(predict.loc2$newy, ymat))] <-
+      scall.pred - 1
> scall.semat <- matrix(NA, length(xmat),
+      length(ymat))
> scall.semat[cbind(match(predict.loc2$newx,
+      xmat), match(predict.loc2$newy, ymat))] <-
+      scallops.pkrige2$se.fit
```

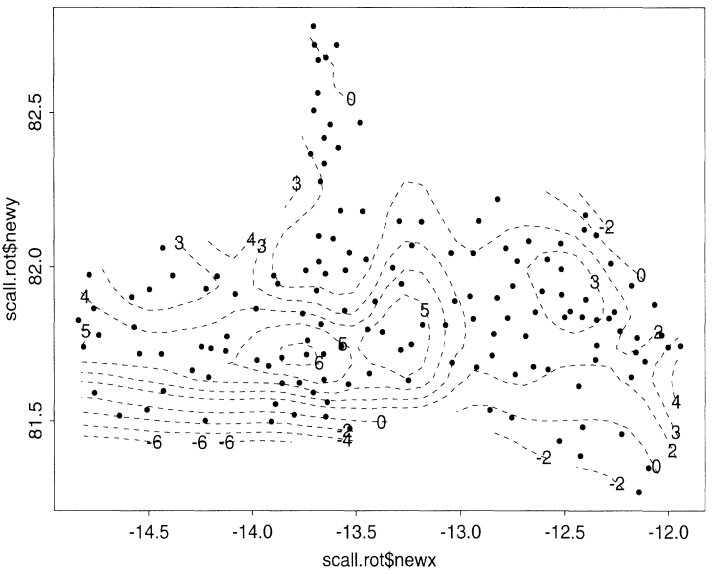

FIGURE 4.23. Contour plot of kriging predictions for logged scallops total catch.

2. Plot the original sampling locations. Overlay contours based on kriging predictions and the spatial loess model. Plot contours for the relative standard errors.

```
> plot(scall.rot$newx, scall.rot$newy)
> contour(xmat,ymat,scall.predmat,
+      levels=c(-6,-4,-2,0,2,3,4,5,6),
+      add=T,lty=3)
> plot(scall.rot$newx, scall.rot$newy)
> contour(xmat,ymat,scall.semat)
```

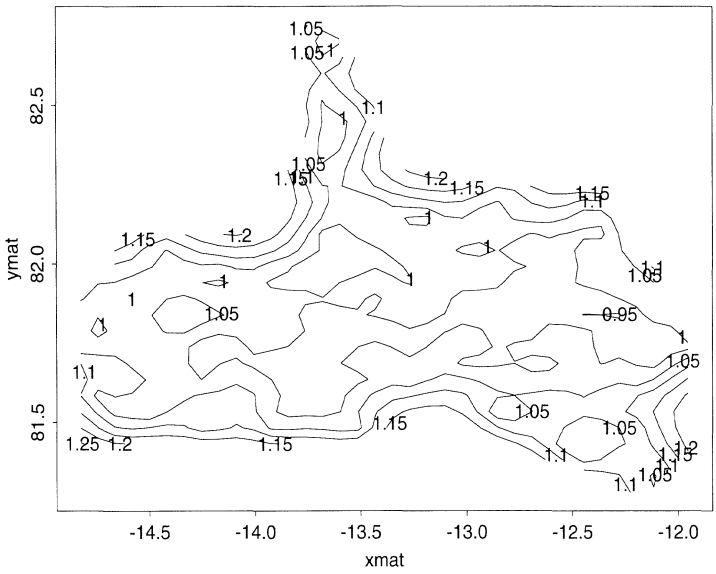

FIGURE 4.24. Contour plot of relative kriging prediction standard errors for logged scallops total catch.

The plots are displayed in figures 4.23 and 4.24.

4.3.2 Universal Kriging

Universal kriging is performed in S+SPATIALSTATS using the same functions as for ordinary kriging; namely, `krige` and `predict.krige`. The difference is in the formula specification for `krige`. Since universal kriging simultaneously fits a trend model to the data, the formula must contain a specification for a polynomial trend surface.

Universal kriging requires knowledge of both a trend model and a variogram or covariance model for the data. The variogram cannot be correctly estimated if there is a trend and the trend cannot be estimated correctly by standard methods if there is spatial correlation. Often an iterative approach is used; initial covariance and trend models are selected and universal kriging is run. The residuals from the trend surface (`resid(obj)`, where `obj` is an object returned from `krige`) can be examined to see if the trend model is correctly specified. The variogram of these residuals can also be used to refine the covariance model. Universal kriging is then run again with the refinements to the model.

In some situations, additional information may allow estimation of the variogram from subsets of the data. With the appropriate assumptions,

this model can then be used in the universal kriging of the entire data set. This is illustrated in the example below.

The use of universal kriging is demonstrated using the coal ash data first introduced in section 3.2.1. Exploratory data analysis showed an apparent trend in the east-west direction. Earlier in this chapter (section 4.1.3) the trend was removed using median polishing. Resulting directional variograms showed essentially no remaining spatial correlation in the east-west direction. As such, the spatial correlation is modeled using a spherical variogram fitted to the empirical variogram in the north-south direction. The parameters were estimated using nonlinear least squares in section 4.2.2. The trend model is specified based on the observed trend in the east-west direction (Cressie, 1986). Perform universal kriging of the coal ash data as follows:

```
> coal.krige <- krige(coal ~ loc(x, y) + x + x^2,
+      data = coal.ash, covfun = spher.cov,
+      range = 4.31, sill = 0.14, nugget = 0.89)
> coal.predict <- predict(coal.krige)
```

View a summary of the `coal.krige` object as follows:

```
> coal.krige
Call:
krige(formula = coal ~ loc(x, y) + x + x^2,
        data = coal.ash, covfun = spher.cov,
        range = 4.31, sill = 0.14, nugget = 0.89)

Coefficients:
 constant         x         x^2
 9.633667 -1.30365 -0.1383046

Number of observations: 208
```

In addition to a summary of the call, estimates for the coefficients of the fitted surface are displayed.

Plot the surface based on the kriging predictions using the Trellis graphics `wireframe` function:

```
> wireframe(fit ~ x * y, data = coal.predict,
+      screen = list(z = 300, x = -60, y = 0),
+      drape = T)
```

The resulting surface is shown in figure 4.25. A similarly produced wire-frame plot of the kriging prediction standard errors, using `se.fit`, is shown in figure 4.26.

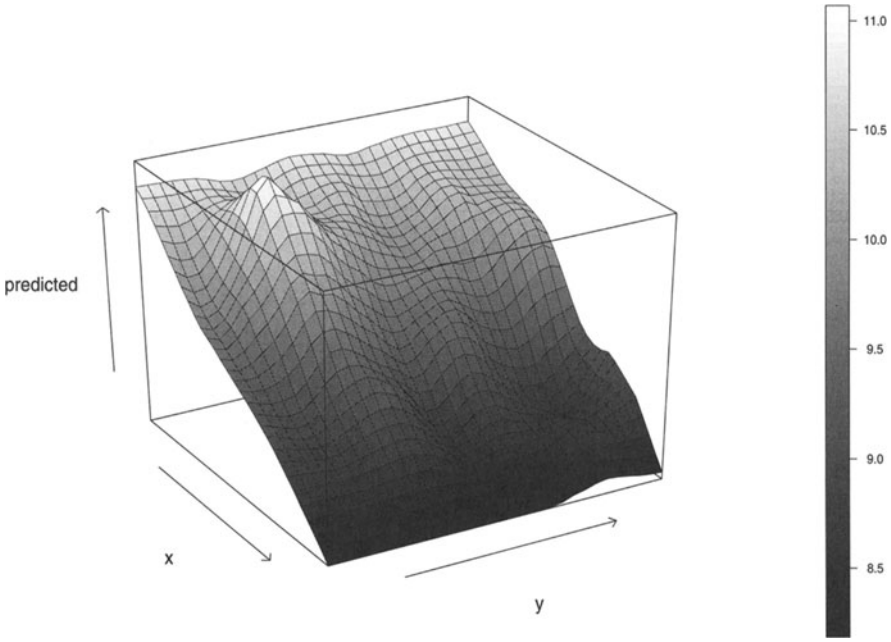

FIGURE 4.25. Surface plot of percent coal ash based on universal kriging predictions.

The modeled trend in the east-west direction is clear from the plot of the predicted surface. The plot of the standard errors reflects the shape of the sampled region; areas on the edges of the prediction grid have higher standard errors since the prediction locations are further away from sampled locations. One might consider kriging over a different prediction region, as was discussed for the scallops data kriging example in section 4.3.1.

4.4 Simulating Geostatistical Data

Simulated data can be useful for examining sampling strategies, as well as for evaluating various prediction methods.

In S+SpatialStats geostatistical data with known autocorrelation can be generated using the function `rfsim`. For example, to simulate a random field on a 20×20 grid, proceed as follows:

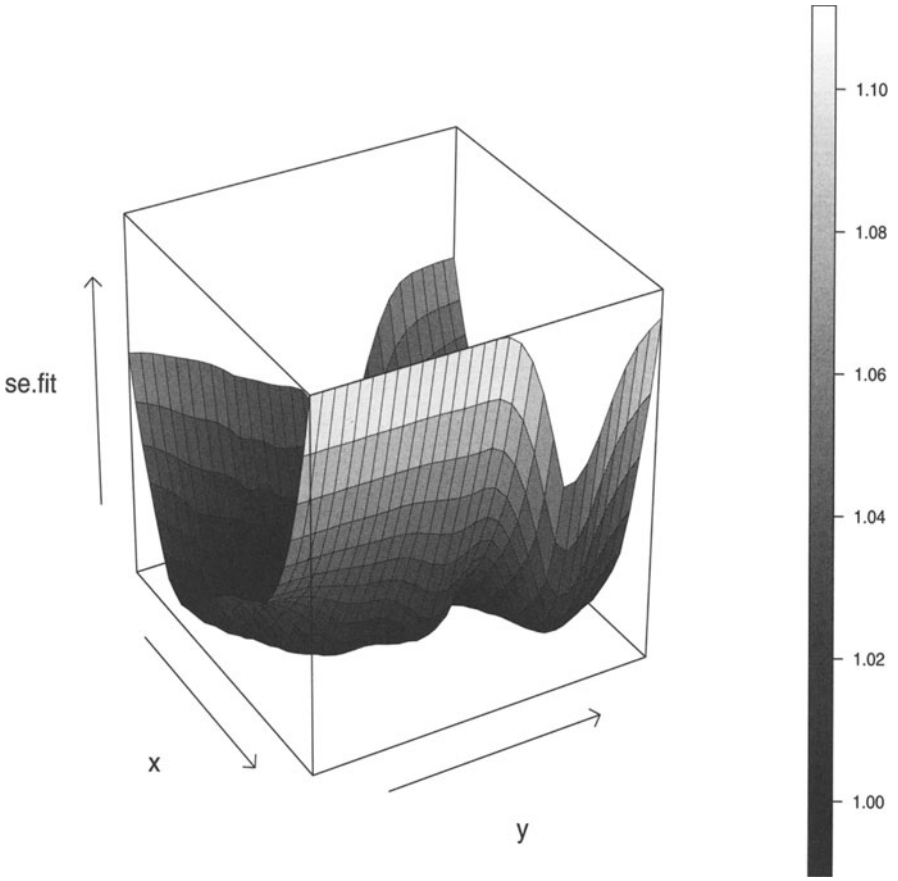

FIGURE 4.26. Surface plot of universal kriging prediction standard errors.

```
> xy20 <- expand.grid(x=seq(0.5,10,len=20),
+       y=seq(0.5,10,len=20))
> z.exp <- rfsim(xy20,covfun=exp.cov,range=2)
```

The function `expand.grid` returns a 400×2 data frame of the spatial locations over the grid. The spatial locations are then input into `rfsim`, along with a model for the covariance structure. By default, the output is a Gaussian random field. The values in *z.exp* are autocorrelated with an exponential distanced based covariance. Figure 4.27 shows the simulated surface and autocorrelation structure, plotted as follows:

```
> par(mfrow=c(2,1))
> persp(unique(xy20$x),unique(xy20$y),
+       matrix(z.exp,20))
```

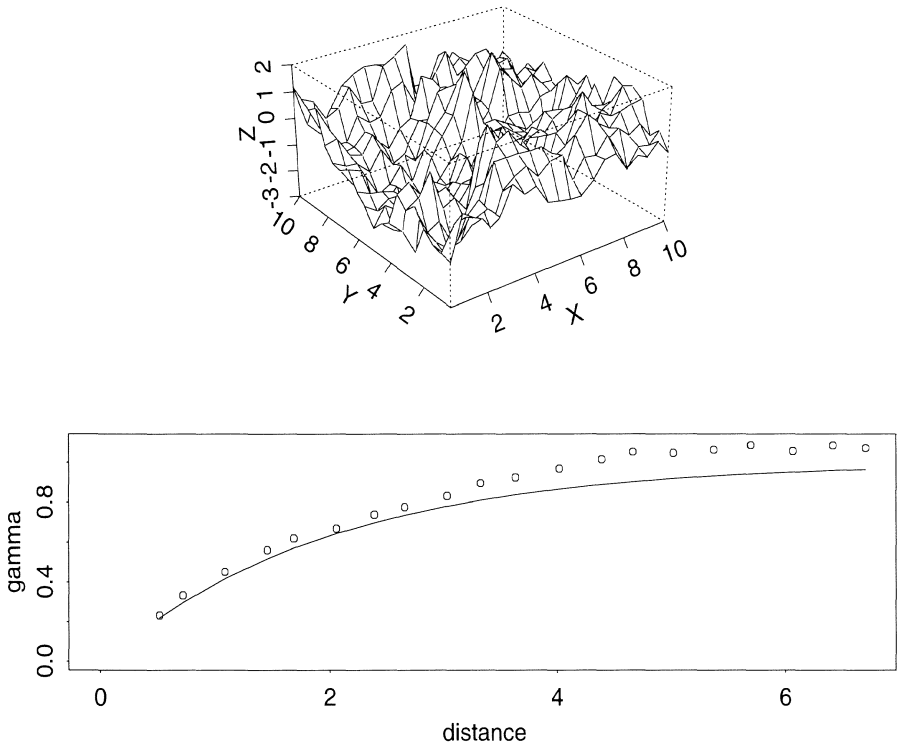

FIGURE 4.27. Perspective plot of the simulated surface (top), the empirical variogram based on the simulated data (open circles, bottom), and the theoretical autocorrelation structure used for the simulation (line, bottom).

```
> z.var <- variogram(z.exp~loc(xy20$x,xy20$y))
> plot(z.var,ylim=c(0,1.1))
> lines(z.var$distance,exp.vgram(
+       z.var$distance,range=2))
```

This simulation represents only one realization of a Gaussian random field with an exponential distanced based covariance.