Chapter 12

Spatial Statistics

12.1 Introduction

We include this final chapter to illustrate an area of data analysis where the methods of computational statistics can be applied. We do not cover this topic in great detail, but we do present some of the areas in spatial statistics that utilize the techniques discussed in the book. These methods include exploratory data analysis and visualization (see Chapter 5), kernel density estimation (see Chapter 8), and Monte Carlo simulation (see Chapter 6).

What Is Spatial Statistics?

*Spatial statistics* is concerned with statistical methods that explicitly consider the spatial arrangement of the data. Most statisticians and engineers are familiar with time-series data, where the observations are measured at discrete time intervals. We know there is the possibility that the observations that come later in the series are dependent on earlier values. When analyzing such data, we might be interested in investigating the temporal data process that generated the data. This can be thought of as an unobservable curve (that we would like to estimate) that is generated in relation to its own previous values.

Similarly, we can view spatial data as measurements that are observed at discrete locations in a two-dimensional region. As with time series data, the observations might be spatially correlated (in two dimensions), which should be accounted for in the analysis.

Bailey and Gatrell [1995] sum up the definition and purpose of spatial statistics in this way:

> observational data are available on some process operating in space and methods are sought to describe or explain the behaviour of this process and its possible relationship to other spatial phenomena. The object of the analysis is to increase our basic understanding of the process, assess the evidence in favour of various hypotheses concerning it, or possibly to predict values
in areas where observations have not been made. The data with which we are concerned constitute a sample of observations on the process from which we attempt to infer its overall behaviour. [Bailey and Gatrell, 1995, p. 7]

Types of Spatial Data

Typically, methods in spatial statistics fall into one of three categories that are based on the type of spatial data that is being analyzed. These types of data are called: point patterns, geostatistical data, and lattice data. The locations of the observations might be referenced as points or as areal units. For example, point locations might be designated by latitude and longitude or by their $x$ and $y$ coordinates. Areal locations could be census tracts, counties, states, etc.

Spatial point patterns are data made up of the location of point events. We are interested in whether or not their relative locations represent a significant pattern. For example, we might look for patterns such as clustering or regularity. While in some point-pattern data we might have an attribute attached to an event, we are mainly interested in the locations of the events. Some examples where spatial statistics methods can be applied to point patterns are given below.

- We have a data set representing the location of volcanic craters in Uganda. It shows a trend in a north-easterly direction, possibly representing a major fault. We want to explore and model the distribution of the craters using methods for analyzing spatial point patterns.
- In another situation, we have two data sets showing thefts in the Oklahoma City area in the 1970’s. One data set corresponds to those committed by Caucasian offenders, and one data set contains information on offences by African-Americans. An analyst might be interested in whether there is a difference in the pattern of offences committed by each group of offenders.
- Seismologists have data showing the distribution of earthquakes in a region. They would like to know if there is any pattern that might help them make predictions about future earthquakes.
- Epidemiologists collect data on where diseases occur. They would like to determine any patterns that might indicate how the disease is passed to other individuals.

With geostatistical data (or spatially continuous data), we have a measurement attached to the location of the observed event. The locations can vary continuously throughout the spatial region, although in practice, measurements (or attributes) are taken at only a finite number of locations. We are not necessarily interested in the locations themselves. Instead, we want to understand and model the patterns in the attributes, with the goal of using

© 2002 by Chapman & Hall/CRC
the model to predict values of the variable at locations where measurements were not taken. Some examples of geostatistical data analysis include the following:

- Rainfall is recorded at various points in a region. These data could be used to model the rainfall over the entire region.
- Geologists take ore samples at locations in a region. They would like to use these data to estimate the extent of the mineral deposit over the entire region.
- Environmentalists measure the level of a pollutant at locations in a region with the goal of using these data to model and estimate the level of pollutant at other locations in the region.

The third type of spatial data is called **lattice data**. These data are often associated with areas that can be regularly or irregularly spaced. The objective of the analysis of lattice data is to model the spatial pattern in the attributes associated with the fixed areas. Some examples of lattice data are:

- A sociologist has data that comprises socio-economic measures for regions in China. The goal of the analysis might be to describe and to understand any patterns of inequality between the areas.
- Market analysts use socio-economic data from the census to target a promising new area to market their products.
- A political party uses data representing the geographical voting patterns in a previous election to determine a campaign schedule for their candidate.

**Spatial Point Patterns**

In this text, we look at techniques for analyzing spatial point patterns only. A spatial point pattern is a set of point locations \( s_1, \ldots, s_n \) in a study region \( R \). Each point location \( s_i \) is a vector containing the coordinates of the \( i \)-th event,

\[
 s_i = \begin{bmatrix} s_{i1} \\ s_{i2} \end{bmatrix}
\]

The term *event* can refer to any spatial phenomenon that occurs at a point location. For example, events can be locations of trees growing in a forest, positions of cells in tissue or the incidence of disease at locations in a community. Note that the scale of our study affects the reasonableness of the assumption that the events occur at point locations.

In our analysis of spatial point patterns, we might have to refer to other locations in the study region \( R \), where the phenomenon was not observed.
We need a way to distinguish them from the locations where observations were taken, so we refer to these other locations as \textit{points} in the region.

At the simplest level, the data we are analyzing consist only of the coordinate locations of the events. As mentioned before, they could also have an attribute or variable associated with them. For example, this attribute might be the date of onset of the disease, the species of tree that is growing, or the type of crime.

This type of spatial data is sometimes referred to as a \textsc{marked point pattern}. In our treatment of spatial point patterns, we assume that the data represent a \textsc{mapped point pattern}. This is one where all relevant events in the study region $R$ have been measured. The study region $R$ can be any shape. However, edge effects can be a problem with many methods in spatial statistics.

We describe the ramifications of edge effects as they arise with the various techniques. In some cases, edge effects are handled by leaving a specified guard area around the edge of the study region, but still within $R$. The analysis of point patterns is sensitive to the definition of $R$, so one might want to perform the analysis for different guard areas and/or different study regions.

One way we can think of spatial point patterns is in terms of the number of events occurring in an arbitrary sub-region of $R$. We denote the number of events in a sub-region $A$ as $Y(A)$. The spatial process is then represented by the random variables $Y(A), A \subset R$. Since we have a random process, we can look at the behavior in terms of the first-order and second-order properties.

These are related to the expected value (i.e., the mean) and the covariance [Bailey and Gatrell, 1995]. The mean and the covariance of $Y(A)$ depend on the number of events in arbitrary sub-regions $A$, and they depend on the size of the areas and the study region $R$. Thus, it is more useful to look at the first- and second-order properties in terms of the limiting behavior per unit area.

The first-order property is described by the intensity $\lambda(s)$. The intensity is defined as the mean number of events per unit area at the point $s$. Mathematically, the intensity is given by

$$\lambda(s) = \lim_{ds \to 0} \left\{ \frac{E[Y(ds)]}{ds} \right\},$$  \hspace{1cm} (12.1)

where $ds$ is a small region around the point $s$, and $ds$ is its area. If it is a \textsc{stationary point process}, then Equation 12.1 is a constant over the study region. We can then write the intensity as

$$E[Y(A)] = \lambda A,$$  \hspace{1cm} (12.2)

where $A$ is the area of the sub-region, and $\lambda$ is the value of the intensity.

To understand the second-order properties of a spatial point process, we need to look at the number of events in pairs of sub-regions of $R$. The second-order property reflects the spatial dependence in the process. We describe
this using the second-order intensity $\gamma(s_i, s_j)$. As with the intensity, this is defined using the events per unit area, as follows,

$$
\gamma(s_i, s_j) = \lim_{d_{s_i}, d_{s_j} \to 0} \left\{ \frac{E[Y(d_{s_i})Y(d_{s_j})]}{d_{s_i}d_{s_j}} \right\}.
$$

(12.3)

If the process is stationary, then $\gamma(s_i, s_j) = \gamma(s_i - s_j)$. This means that the second-order intensity depends only on the vector difference of the two points. The process is said to be **second-order** and **isotropic** if the second-order intensity depends only on the distance between $s_i$ and $s_j$. In other words, it does not depend on the direction.

**Complete Spatial Randomness**

The benchmark model for spatial point patterns is called **complete spatial randomness** or CSR. In this model, events follow a homogeneous Poisson process over the study region. The definition of CSR is given by the following [Diggle, 1983):

1. The intensity does not vary over the region. Thus, $Y(A)$ follows a Poisson distribution with mean $\lambda A$, where $A$ is the area of $A$ and $\lambda$ is constant.
2. There are no interactions between the events. This means that, for a given $n$, representing the total number of events in $R$, the events are uniformly and independently distributed over the study region.

In a CSR process, an event has the same probability of occurring at any location in $R$, and events neither inhibit nor attract each other. The methods covered in this chapter are mostly concerned with discovering and modeling departures from the CSR model, such as regularity and clustering. Realizations of these three types of spatial point processes are shown in Figures 12.1 through 12.3, so the reader can understand the differences between these point patterns.

In Figure 12.1, we have an example of a spatial point process that follows the CSR model. Note that there does not appear to be systematic regularity or clustering in the process. The point pattern displayed in Figure 12.2 is a realization of a cluster process, where the clusters are obviously present. Finally, in Figure 12.3, we have an example of a spatial point process that exhibits regularity.

In this chapter, we look at methods for exploring and for analyzing spatial point patterns only. We follow the treatment of this subject that is given in Bailey and Gatrell [1995]. In keeping with the focus of this text, we emphasize the simulation and computational approach, rather than the theoretical. In the next section, we look at ways to visualize spatial point patterns using the
In this figure, we show a realization from a CSR point process.

Here we have an example of a spatial point process that exhibits clustering.

This spatial point process exhibits regularity.
Chapter 12: Spatial Statistics

Section 12.3 contains information about exploring spatial point patterns and includes methods for estimating first-order and second-order properties of the underlying point process. In Section 12.4, we discuss how to model the observed spatial pattern, with an emphasis on comparing the observed pattern to one that is completely spatially random. Finally, in Section 12.5, we offer some other models for spatial point patterns and discuss how to simulate data from them.

12.2 Visualizing Spatial Point Processes

The most intuitive way to visualize a spatial point pattern is to plot the data as a dot map. A dot map shows the region over which the events are observed, with the events shown using plotting symbols (usually points). When the boundary region is not part of the data set, then the dot map is the same as a scatterplot.

We mentioned briefly in Section 12.1 that some point patterns could have an attribute attached to each event. One way to visualize these attributes is to use different colors or plotting symbols that represent the values of the attribute. Another option is to plot text that specifies the attribute value at the event locations. For example, if the data represent earthquakes, then one could plot the level of the quake at each event location. However, this can be hard to interpret and gets cluttered if there are a lot of observations. Plotting this type of scatterplot is easily done in MATLAB using the text function. Its use will be illustrated in the exercises.

In some cases, the demographics of the population (e.g., number of people, age, income, etc.) over the study region is important. For example, if the data represent incidence of disease, then we might expect events to be clustered in regions of high population density. One way to visualize this is to combine the dot map with a surface representing the attribute, similar to what we show in Example 12.4.

We will be using various data sets in this chapter to illustrate spatial statistics for point patterns. We describe them in the next several examples and show how to construct dot maps and boundaries in MATLAB. All of these data sets are analyzed in Bailey and Gatrell [1995].

Example 12.1

In this first example, we look at data comprised of the crater centers of 120 volcanoes in west Uganda [Tinkler, 1971]. We see from the dot map in Figure 12.4 that there is an indication of a regional trend in the north-easterly direction. The data are contained in the file uganda, which contains the

© 2002 by Chapman & Hall/CRC
boundary as well as the event locations. The following MATLAB code shows how to obtain a dot map.

```matlab
load uganda
% This loads up x and y vectors corresponding
% to point locations.
% It also loads up a two column matrix
% containing the vertices to the region.
% Plot locations as points.
plot(x,y,'.k')
hold on
% Plot boundary as line.
plot(ugpoly(:,1),ugpoly(:,2),'k')
hold off
title('Volcanic Craters in Uganda')
```

![Volcanic Craters in Uganda](image)

**FIGURE 12.4**
This dot map shows the boundary region for volcanic craters in Uganda.

**Example 12.2**
Here we have data for the locations of homes of juvenile offenders living in a housing area in Cardiff, Wales [Herbert, 1980] in 1971. We will use these data in later examples to determine whether they show evidence of clustering or spatial randomness. These data are in the file called `cardiff`. When this is
loaded using MATLAB, one also obtains a polygon representing the boundary. The following MATLAB commands construct the dot map using a single call to the `plot` function. The result is shown in Figure 12.5.

```matlab
load cardiff
% This loads up x and y vectors corresponding
% to point locations. It also loads up a two
% column matrix containing the vertices
% to the region.
% Plot locations as points and boundary as line.
% Note: can do as one command:
plot(x,y,'.k',cardpoly(:,1),cardpoly(:,2),'k')
```

Example 12.3

These data are the locations where thefts occurred in Oklahoma City in the late 1970’s [Bailey and Gatrell, 1995]. There are two data sets: 1) `okwhite` contains the data for Caucasian offenders and 2) `okblack` contains the event locations for thefts committed by African-American offenders. Unlike the previous data sets, these do not have a specific boundary associated with them. We show in this example how to get a boundary for the `okwhite` data.
using the MATLAB function \texttt{convhull}. This function returns a set of indices to events in the data set that lie on the convex hull of the locations.

\begin{verbatim}
load okwhite
% Loads up two vectors: okwhx, okwhy
% These are event locations for the pattern.
% Get the convex hull.
K = convhull(okwhx, okwhy);
% K contains the indices to points on the convex hull.
% Get the events.
cvh = [okwhx(K), okwhy(K)];
plot(okwhx, okwhy, 'k.', cvh(:,1), cvh(:,2), 'k')
title('Location of Thefts by Caucasian Offenders')
\end{verbatim}

A plot of these data and the resulting boundary are shown in Figure 12.6. We show in one of the exercises how to use a function called \texttt{csgetregion} (included with the Computational Statistics Toolbox) that allows the user to interactively set the boundary.

\textbf{FIGURE 12.6}
This shows the event locations for locations of thefts in Oklahoma City that were committed by Caucasians. The boundary is the convex hull.
12.3 Exploring First-order and Second-order Properties

In this section, we look at ways to explore spatial point patterns. We see how to apply the density estimation techniques covered in Chapter 8 to estimate the intensity or first-order property of the spatial process. The second-order property can be investigated by using the methods of Chapter 5 to explore the distributions of nearest neighbor distances.

**Estimating the Intensity**

One way to summarize the events in a spatial point pattern is to divide the study region into sub-regions of equal area. These are called *quadrats*, which is a name arising from the historical use of square sampling areas used in field sampling. By counting the number of events falling in each of the quadrats, we end up with a histogram or frequency distribution that summarizes the spatial pattern. If the quadrats are non-overlapping and completely cover the spatial region of interest, then the quadrat counts convert the point pattern into area or lattice data. Thus, the methods appropriate for lattice data can be used.

To get an estimate of intensity, we divide the study region using a regular grid, count the number of events that fall into each square and divide each count by the area of the square. We can look at various plots, as shown in Example 12.4, to understand how the intensity of the process changes over the study region.

Note that if edge effects are ignored, then the other methods in Chapter 8, such as frequency polygons or average shifted histograms can also be employed to estimate the first-order effects of a spatial point process.

Not surprisingly, we can apply kernel estimation to get an estimate of the intensity that is smoother than the quadrat method. As before, we let \( s \) denote a point in the study region \( R \) and \( s_1, ..., s_n \) represent the event locations. Then an estimate of the intensity using the kernel method is given by

\[
\hat{\lambda}_h(s) = \frac{1}{\delta_h(s)} \sum_{i=1}^{n} \frac{1}{h^2} k\left(\frac{s - s_i}{h}\right),
\]

(12.4)

where \( k \) is the kernel and \( h \) is the bandwidth. The kernel is a bivariate probability density function as described in Chapter 8. In Equation 12.4, the edge-correction factor is

\[
\delta_h(s) = \int_{R} \frac{1}{h^2} k\left(\frac{s - u}{h}\right) du.
\]

(12.5)
Equation 12.5 represents the volume under the scaled kernel centered on $s$ which is inside the study region $R$. As with the quadrat method, we can look at how $\hat{\lambda}(s)$ changes to gain insight about the intensity of the point process.

The same considerations, as discussed in Chapter 8, regarding the choice of the kernel and the bandwidth apply here. An overly large $h$ provides an estimate that is very smooth, possibly hiding variation in the intensity. A small bandwidth might indicate more variation than is warranted, making it harder to see the overall pattern in the intensity. A recommended choice for the bandwidth is $h = 0.68n^{-0.2}$, when $R$ is the unit square [Diggle, 1981]. This value could be appropriately scaled for the size of the actual study region.

Bailey and Gatrell [1995] recommend the following quartic kernel

$$
    k(u) = \frac{3}{\pi} \left(1 - u^T u\right)^2 \quad u^T u \leq 1. 
$$

When this is substituted into Equation 12.4, we have the following estimate for the intensity

$$
    \hat{\lambda}_h(s) = \sum_{d_i \leq h} \frac{3}{\pi h^2} \left(1 - \frac{d_i^2}{h^2}\right)^2, 
$$

where $d_i$ is the distance between point $s$ and event location $s$, and the correction for edge effects $\delta_h(s)$ has, for simplicity, not been included.

**Example 12.4**

In this example, we apply the kernel method as outlined above to estimate the intensity of the `uganda` data. We include a function called `csintenkern` that estimates the intensity of a point pattern using the quartic kernel. For simplicity, this function ignores edge effects. The following MATLAB code shows how to apply this function and how to plot the results. Note that we set the window width to $h = 220$. Other window widths are explored in the exercises. First, we load the data and call the function. The output variable `lamhat` contains the values of the estimated intensity.

```matlab
load uganda
X = [x,y];
h = 220;
[xl,yl,lamhat] = csintenkern(X,ugpoly,h);
```

We use the `pcolor` function to view the estimated intensity. To get a useful color map, we use an inverted gray scale. The estimated intensity is shown in Figure 12.7, where the ridge of higher intensity is visible.

```matlab
pcolor(xl,yl,lamhat)
map = gray(256);
```
% Flip the colormap so zero is white and max is black.
map = flipud(map);
colormap(map)
shading flat
hold on
plot(ugpoly(:,1),ugpoly(:,2),'k')
hold off

Of course, one could also plot this as a surface. The MATLAB code we provide below shows how to combine a surface plot of the intensity with a dot map below. The axes can be rotated using the toolbar button or the \texttt{rotate3d} command to look for an interesting viewpoint.

% First plot the surface.
surf(xl,yl,lamhat)
map = gray(256);
map = flipud(map);
colormap(map)
shading flat
% Now plot the dot map underneath the surface.
X(:,3) = -max(lamhat(:))*ones(length(x),1);
ugpoly(:,3) = -max(lamhat(:))...
ones(length(ugpoly(:,1)),1);
hold on
plot3(X(:,1),X(:,2),X(:,3),'.')
plot3(ugpoly(:,1),ugpoly(:,2),ugpoly(:,3),'k')
hold off
axis off
grid off

The combination plot of the intensity surface with the dot map is shown in Figure 12.8.

![Intensity surface with dot map](image)

**Figure 12.8**
This shows the kernel estimate of the intensity along with a dot map.

**Estimating the Spatial Dependence**

We now turn our attention to the problem of exploring the second-order properties of a spatial point pattern. These exploratory methods investigate the second-order properties by studying the distances between events in the study region $R$. We first look at methods based on the nearest neighbor distances between events or between points and events. We then discuss an alternative approach that summarizes the second-order effects over a range of distances.

**Nearest Neighbor Distances - $G$ and $F$ Distributions**

The nearest neighbor event-event distance is represented by $W$. This is defined as the distance between a randomly chosen event and the nearest neighboring event. The nearest neighbor point-event distance, denoted by $X$, is the distance between a randomly selected point in the study region and the
nearest event. Note that nearest neighbor distances provide information at small physical scales, which is a reasonable approach if there is variation in the intensity over the region $R$.

It can be shown [Bailey and Gatrell, 1995; Cressie 1993] that, if the CSR model holds for a spatial point process, then the cumulative distribution function for the nearest neighbor event-event distance $W$ is given by

$$ G(w) = P(W \leq w) = 1 - e^{-\lambda \pi w^2}, \quad (12.8) $$

for $w \geq 0$. The cumulative distribution function for the nearest neighbor point-event distance $X$ is

$$ F(x) = P(X \leq x) = 1 - e^{-\lambda \pi x^2}, \quad (12.9) $$

with $x \geq 0$.

We can explore the second-order properties of a spatial point pattern by looking at the observed cumulative distribution function of $X$ or $W$. The empirical cumulative distribution function for the event-event distances $W$ is given by

$$ \hat{G}(w) = \frac{\#(w \leq w)}{n}. \quad (12.10) $$

Similarly, the empirical cumulative distribution function for the point-event distances $X$ is

$$ \hat{F}(x) = \frac{\#(x \leq x)}{m}, \quad (12.11) $$

where $m$ is the number of points randomly sampled from the study region.

A plot of $\hat{G}(w)$ and $\hat{F}(x)$ provides possible evidence of inter-event interactions. If there is clustering in the point pattern, then we would expect a lot of short distance neighbors. This means that $\hat{G}(w)$ would climb steeply for smaller values of $w$ and flatten out as the distances get larger. On the other hand, if there is regularity, then there should be more long distance neighbors and $\hat{G}(w)$ would be flat at small distances and climb steeply at larger $w$ or $x$. When we examine a plot of $\hat{F}(x)$, the opposite interpretation holds. For example, if there is an excess of long distances values in $\hat{F}(x)$, then that is evidence for clustering.

We could also plot $\hat{G}(w)$ against $\hat{F}(x)$. If the relationship follows a straight line, then this is evidence that there is no spatial interaction. If there is clustering, then we expect $\hat{G}(w)$ to exceed $\hat{F}(x)$, with the opposite situation occurring if the point pattern exhibits regularity.
From Equation 12.8, we can construct a simpler display for detecting departures from CSR. Under CSR, we would expect a plot of

$$\left\{ -\log \left( 1 - \hat{G}(w) \right) \right\}^{1/2} \over \left( \hat{\lambda} \pi \right)$$

(12.12)

everson $w$ to be a straight line. In Equation 12.12, we need a suitable estimate for the intensity $\hat{\lambda}$. One possibility is to use $\hat{\lambda} = n/r$, where $r$ is the area of the study region $R$.

So far, we have not addressed the problem of edge effects. Events near the boundary of the region $R$ might have a nearest neighbor that is outside the boundary. Thus, the nearest neighbor distances near the boundary might be biased. One possible solution is to have a guard area inside the perimeter of $R$. We do not compute nearest neighbor distances for points or events in the guard area, but we can use events in the guard area in computing nearest neighbors for points or events inside the rest of $R$. Other solutions for making corrections are discussed in Bailey and Gatrell [1995] and Cressie [1993].

Example 12.5

The data in $bodmin$ represent the locations of granite tors on Bodmin Moor [Pinder and Witherick, 1977; Upton and Fingleton, 1985]. There are 35 locations, along with the boundary. The $x$ and $y$ coordinates for the locations are stored in the $x$ and $y$ vectors, and the vertices for the region are given in $bodpoly$. The reader is asked in the exercises to plot a dot map of these data. In this example, we use the event locations to illustrate the nearest neighbor distribution functions $\hat{G}(w)$ and $\hat{F}(x)$. First, we show how to get the empirical distribution function for the event-event nearest neighbor distances.

```
load bodmin
% Loads data in x and y and boundary in bodpoly.
% Get the Ghat function first and plot.
X = [x,y];
w = 0:.1:10;
n = length(x);
nw = length(w);
ghat = zeros(1,nw);
% The G function is the nearest neighbor
% distances for each event.
ghat = zeros(1,nw);
% Find the distances for all points.
dist = pdist(X);
% Convert to a matrix and put large
% numbers on the diagonal.
D = diag(realmax*ones(1,n)) + squareform(dist);
% Find the smallest distances in each row or col.
```
mind = min(D);
% Now get the values for ghat.
for i = 1:nw
    ind = find(mind<=w(i));
    ghat(i) = length(ind);
end
ghat = ghat/n;

To see whether there is evidence for clustering or regularity, we plot $\hat{G}(w)$ using the following commands.

% Plot the Ghat as a function of w. Shows evidence
% of clustering.
figure,plot(w,ghat,'k')
axis([0 10 0 1.1])
xlabel('Event-Event Distances - w'),ylabel('Ghat')

We see from Figure 12.9, that the curve climbs steeply at small values of $w$, providing possible evidence for clustering. This indicates that there are many small event-event distances, which is what we would expect for clustering. The reader is asked to explore this further in the exercises by plotting the expression in Equation 12.12 versus $w$. Next, we determine $F(x)$. First we find the nearest neighbor distances for $m = 75$ randomly selected points in the study region.

![Graph showing empirical distribution function](image)

**Figure 12.9**
This is the empirical distribution function for the event-event nearest neighbor distances for the bodmin data. This provides possible evidence for clustering.
\texttt{xx = w; }  \\
\texttt{m = 75; }  \\
\texttt{nx = length(xx); }  \\
\texttt{fhat = zeros(1,nx);}  \\
\texttt{mind = zeros(1,m);} \% one for each point \texttt{m}  \\
\texttt{xt = [0 0; X]; }  \\
\% The \texttt{F} function is the nearest neighbor distances for \\
\% randomly selected points. Generate a point, find its \\
\% closest event.  \\
\texttt{for i = 1:m}  \\
\texttt{ \% Generate a point in the region.}  \\
\texttt{ [xt(1,1), xt(1,2)] = csbinproc(bodpoly(:,1),...}  \\
\texttt{ bodpoly(:,2), 1);}  \\
\texttt{ \% Find the distances to all events.}  \\
\texttt{ dist = pdist(xt);}  \\
\texttt{ \% The first \texttt{n} in \texttt{dist} are the distances}  \\
\texttt{ \% between the point(first row) and all the events.}  \\
\texttt{ \% Find the smallest here.}  \\
\texttt{ mind(i) = min(dist(1:n)); }  \\
\texttt{end}  \\

Now that we have the nearest neighbor distances, we can find the empirical \\
distribution function, as follows.  \\
\texttt{\% Now get the values for \texttt{fhat}.}  \\
\texttt{for i = 1:nx}  \\
\texttt{ ind = find(mind<=xx(i)); }  \\
\texttt{ fhat(i) = length(ind);}  \\
\texttt{end}  \\
\texttt{fhat = fhat/m;}  \\

We plot the empirical distribution function \( \hat{F}(x) \) in Figure 12.10, where it also \\
seems to provide evidence for the cluster model.  \\
\[ \]  \\

\textit{K-Function}  \\
The empirical cumulative distribution functions \( \hat{G}(w) \) and \( \hat{F}(x) \) use \\
distances to the nearest neighbor, so they consider the spatial point pattern over \\
the smallest scales. It would be useful to have some insight about the pattern \\
at several scales. We use an estimate of the \textit{K}-function, which is related to the \\
second-order properties of an isotropic process [Ripley, 1976, 1981]. If the \textit{K}- \\
function is used when there are first-order effects over large scales, then spatial \\
dependence indicated by the \textit{K}-function could be due to first-order effects \\
instead [Bailey and Gatrell, 1995]. If this is the case, the analyst might want to \\
study sub-regions of \( R \) where first-order homogeneity is valid.  \\
The \textit{K}-function is defined as
Chapter 12: Spatial Statistics

\[ K(d) = \lambda^{-1} E[\# \text{ extra events within distance } d \text{ of an arbitrary event}], \]

where \( \lambda \) is a constant representing the intensity over the region and \( E[.] \) denotes the expected value.

An edge corrected estimate for the \( K \)-function is given by the following

\[
\hat{K}(d) = \frac{r}{n^2} \sum_{i \neq j} \frac{I(d_{ij})}{w_{ij}}. \tag{12.13}
\]

In Equation 12.13, \( r \) represents the area of the study region \( R \), \( n \) is the number of events, \( d_{ij} \) is the distance between the \( i \)-th and \( j \)-th events, and \( I_d \) is an indicator function that takes on the value of one if \( d_{ij} \leq d \) and zero otherwise. The \( w_{ij} \) in Equation 12.13 is a correction factor for edge effects. If a circle is centered at event \( i \) and passes through event \( j \), then \( w_{ij} \) is the proportion of the circumference of the circle that is in region \( R \).

The estimated \( K \)-function can be compared to what we would expect if the process that generated the data is completely spatially random. For a CSR spatial point process, the theoretical \( K \)-function is

\[
K(d) = \pi d^2. \tag{12.14}
\]

FIGURE 12.10
This is the empirical distribution function for the point-event distances of the bodmin data.
If our observed process exhibits regularity for a given value of $d$, then we expect that the estimated $K$-function will be less than $\pi d^2$. Alternatively, if the spatial pattern has clustering, then $K(d) > \pi d^2$. Plots of the $\hat{K}(d)$ and $K(d)$ under CSR (Equation 12.14) enable us to explore the second-order properties of the spatial process.

Another approach, based on the $K$-function, is to transform $\hat{K}(d)$ using

$$
\hat{L}(d) = \frac{\hat{K}(d)}{\sqrt{\pi d}} - d.
$$

(12.15)

Peaks of positive values in a plot of $\hat{L}(d)$ would correspond to clustering, with troughs of negative values indicating regularity, for the corresponding scale $d$. Note that with $K(d)$ and $L(d)$, we can explore spatial dependence at a range of scales $d$. The quantity

$$
L(d) \equiv \frac{K(d)}{\sqrt{\pi d}} - d
$$

(12.16)

is called the $L$-function, and Equation 12.15 is an estimate of it.

**Example 12.6**

In this example, we find $\hat{K}(d)$ and $\hat{L}(d)$ for the `cardiff` data set. We provide a function in the Computational Statistics Toolbox called `cskhat` for estimating the $K$-function and illustrate its use below.

```matlab
load cardiff
% Loads data in x and y and region in cardpoly.
% Get the scales or distances for K_hat.
d = 1:30;
X = [x,y];
% Get the estimate of K_hat.
khat = cskhat(X, cardpoly, 1:30);
```

The next commands show how to plot $\hat{K}(d)$ and the theoretical $K$-function for a random process.

```matlab
% Plot the khat function along with the K-function
% under CSR. Shows clustering because $khat$ is above the curve.
plot(d,pi*d.^2,'k',d,khat,'k')
xlabel('Distances - d')
ylabel('K Function')
```

This plot is given in Figure 12.11, where we see possible evidence for clustering, because the observed $K$-function is above the curve corresponding to a
random process. As mentioned previously, we can also plot the function \( \hat{L}(d) \). This is shown in Figure 12.12, where we see clustering at all scales.

% Get the Lhat function.
% Positive peaks - clustering at all of these scales.
% Clustering shown at d = 10, showing possible
% clustering at that scale.

\[
\hat{l}(d) = \sqrt{k(d)/\pi} - d;
\]

plot(d, lhat, 'k')
xlabel('Distances - d')
ylabel('Lhat')

This shows the function \( \hat{K}(d) \) for the \texttt{cardiff} data. Note that it is above the curve for a random process, indicating possible clustering.

### 12.4 Modeling Spatial Point Processes

When analyzing spatial point patterns, we are mainly interested in discovering patterns such as clustering or regularity versus complete spatial randomness. The exploratory methods of the previous section are meant to provide
evidence for a model that might explain the process that generated the spatial point pattern. We now look at ways to use Monte Carlo hypothesis testing to understand the statistical significance of our evidence for departures from CSR. These tests are based on nearest neighbor distances and the $K$-function.

**Nearest Neighbor Distances**

Recall that the theoretical cumulative distribution function (under the CSR model) for the nearest neighbor event-event distance $W$ is given by

$$G(w) = P(W \leq w) = 1 - e^{-\lambda \pi w^2}; \quad w \geq 0,$$  \hspace{1cm} (12.17)

and the cumulative distribution function for the nearest neighbor point-event distance $X$ is

$$F(x) = P(X \leq x) = 1 - e^{-\lambda \pi x^2}; \quad x \geq 0.$$  \hspace{1cm} (12.18)

These distributions can be used to implement statistical hypothesis tests that use summary statistics of the observed nearest neighbor distances. The estimated distributions, $\hat{G}(w)$ or $\hat{F}(x)$, can be plotted against the corre-
sponding theoretical distributions under CSR. If the CSR model is valid for the observed spatial point process, then we would expect these plots to follow a straight line. Equations 12.17 and 12.18 assume that no edge effects are present, so it is important to correct for the edge effects when calculating $G(w)$ and $F(x)$. The reader is referred to Cressie [1993, p. 614] for a description of the edge corrections for $G(w)$ and $F(x)$. As with the exploratory methods described in the previous section, it is difficult to assess the significance of any departure from CSR that is seen in the plots, even though we might suspect such a departure.

In the plots discussed in the previous section, we have to judge the general shape of the curve for $G(w)$ or $F(x)$, which is subjective and not very exact. We now offer another useful way to display these functions. When we plot the empirical distributions for the observed nearest neighbor distances against the theoretical distributions, we expect a straight line, if the point pattern follows a CSR process. In a clustered process, the curve for $F(x)$ would lie below the 45 degree line as shown in Figure 12.13 for the Bodmin Tors data. If the process exhibits regularity, then the empirical distribution function $F(x)$ lies above the line. As before, the opposite interpretation holds for the distribution function $G(w)$.

FIGURE 12.13
This is the empirical point-event nearest neighbor distribution function $F(x)$ for the Bodmin Tors data. Since the curve lies below the 45 degree line, this indicates clustering. Note that edge effects have been ignored.
We now describe simulation techniques that compare the estimated distribution functions with the distribution under CSR, allowing the analyst to assess the significance of any departure from CSR. These methods are particularly useful, because the edge effects are taken care of by the simulation procedure, so explicit corrections do not need to be made. However, we note that edge-corrected statistics may lead to more powerful tests than those that do not incorporate the edge corrections.

In the procedure explained below, we see that edge effects are accounted for because of the following:

1. The estimated distributions \( \hat{G}(w) \) and \( \hat{F}(x) \) are obtained for \( R \) without edge correction.
2. The estimate of the distribution under CSR is obtained via simulation for the particular study region \( R \). In other words, we use a procedure that, for a given \( n \), yields events that are uniformly and independently distributed over the region. See Section 12.5 for more information.

We describe the method as it applies to the point-event distances \( X \), with an analogous approach holding for the event-event distances \( W \). In Example 12.7, we illustrate the procedure as it applies to \( W \) and leave the other as an exercise for the reader. The simulation estimate for \( F(x) \) under CSR is obtained by first generating \( B \) spatial point patterns of size \( n \) that are independently and uniformly distributed over \( R \). The empirical cumulative distribution function is determined for each simulated point pattern, without correcting for edge effects. We denote these by \( \hat{F}_b(x) \), \( b = 1, \ldots, B \). Taking the mean of these functions yields an estimate of the distribution of the point-event nearest neighbor distances for a process under CSR,

\[
\hat{F}_{CSR}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{F}_b(x). \tag{12.19}
\]

Letting \( \hat{F}_{Obs}(x) \) denote the empirical cumulative distribution function for the observed spatial point pattern, we can plot \( \hat{F}_{Obs}(x) \) against \( \hat{F}_{CSR}(x) \). If the data follow the CSR model, then the plot should be a straight line. If the data exhibit clustering, then the plot will be above the line. If regularity is present, then the plot will be below the line.

We can assess the significance of the departure from CSR by constructing upper and lower simulation envelopes. These are given by

\[
U(x) = \max_b \{ \hat{F}_b(x) \}, \tag{12.20}
\]

and
\[ L(x) = \min_{i} \{ \hat{F}_i(x) \} . \]  

(12.21)

The significance of the departure from CSR is found using

\[ P(\hat{F}_{\text{obs}}(x) > U(x)) = P(\hat{F}_{\text{obs}}(x) < L(x)) = \frac{1}{B + 1} . \]  

(12.22)

For example, if we want to detect clustering that is significant at \( \alpha = 0.05 \), then (from Equation 12.22) we need 19 simulations. Adding the upper and lower simulation envelopes to the plot of \( \hat{F}_{\text{obs}}(x) \) against \( \hat{F}_{\text{CSR}}(x) \) enables us to determine the significance of the clustering. If \( \hat{F}_{\text{obs}}(x) \) is below the upper envelope, then the result showing clustering is significant. Note that Equation 12.22 is for a fixed \( x \), so the analyst must look at each point in the curve of \( \hat{F}_{\text{obs}}(x) \). In the exercises, we describe an alternative, more powerful test.

PROCEDURE - MONTE CARLO TEST USING NEAREST NEIGHBOR DISTANCES

1. Obtain the empirical cumulative distribution function using the observed spatial point pattern, \( \hat{F}_{\text{obs}}(x) \) (or \( \hat{G}_{\text{obs}}(w) \)). Do not correct for edge effects.
2. Simulate a spatial point pattern over the study region of size \( n \) from a CSR process.
3. Get the empirical cumulative distribution function \( \hat{F}_b(x) \) (or \( \hat{G}_b(w) \)). Do not correct for edge effects.
4. Repeat steps 2 and 3, \( B \) times, where \( B \) is determined from Equation 12.22.
5. Take the average of the \( B \) distributions using Equation 12.19 to get the estimated distribution of the nearest neighbor distances under CSR, \( \hat{F}_{\text{CSR}}(x) \) (or \( \hat{G}_{\text{CSR}}(w) \)).
6. Find the lower and upper simulation envelopes.
7. Plot \( \hat{F}_{\text{obs}}(x) \) (or \( \hat{G}_{\text{obs}}(w) \)) against \( \hat{F}_{\text{CSR}}(x) \) (or \( \hat{G}_{\text{CSR}}(w) \)).
8. Add plots of the lower and upper simulation envelopes to assess the significance of the test.

Example 12.7

In this example, we show how to implement the procedure for comparing \( \hat{G}_{\text{obs}}(w) \) with an estimate of the empirical distribution function under CSR. We use the \texttt{bodmin} data set, so we can compare this with previous results. First we get \( \hat{G}_{\text{obs}}(w) \).

```matlab
load bodmin
X = [x,y];
% Note that we are using a smaller range
```

© 2002 by Chapman & Hall/CRC
The next step is to simulate from a CSR process over the same region and determine the empirical event-event distribution function for each simulation.

% Get the simulations.
B = 99;
% Each row is a Ghat from a simulated CSR process.
simul = zeros(B,nw);
for b = 1:B
    [xt,yt] = csbinproc(bodpoly(:,1), bodpoly(:,2), nx);
simul(b,:) = csghat([xt,yt],w);
end

We need to take the average of all of the simulations so we can plot these values along the horizontal axis. The average and the envelopes are easily found in MATLAB. The resulting plot is given in Figure 12.14. Note that there does not seem to be significant evidence for departure from the CSR model using the event-event nearest neighbor distribution function $\hat{G}_{\text{Obs}}(w)$.

% Get the average.
ghatmu = mean(simul);
% Get the envelopes.
ghatup = max(simul);
ghatlo = min(simul);
plot(ghatmu,ghatobs,'k',ghatmu,ghatup,'k--',ghatmu,ghatlo,'k--')

K-Function

We can use a similar approach to formally compare the observed $K$-function with an estimate of the $K$-function under CSR. We determine the upper and lower envelopes as follows

$$U(d) = \max_b \{ \hat{K}_b(d) \},$$

and

$$L(d) = \min_b \{ \hat{K}_b(d) \}.$$
The $\hat{K}_n(d)$ are obtained by simulating spatial point patterns of size $n$ events in $R$ under CSR.

Alternatively, we can use the $L$-function to assess departures from CSR. The upper and lower simulation envelopes for the $L$-function are obtained in the same manner. With the $L$-function, the significance of the peaks or troughs (for fixed $d$) can be assessed using

$$P(\hat{L}_{\text{Obs}}(d) > U(d)) = P(\hat{L}_{\text{Obs}}(d) < L(d)) = \frac{1}{B + 1}. \quad (12.25)$$

We outline the steps in the following procedure and show how to implement them in Examples 12.8 and 12.9.

**PROCEDURE - MONTE CARLO TEST USING THE K-FUNCTION**

1. Estimate the $K$-function using the observed spatial point pattern to get $\hat{K}_{\text{Obs}}(d)$.
2. Simulate a spatial point pattern of size $n$ over the region $R$ from a CSR process.
3. Estimate the $K$-function using the simulated pattern to get $\hat{K}_b(d)$.
4. Repeat steps 2 and 3, $B$ times.
6. Plot $\hat{K}_{\text{Obs}}(d)$ and the simulation envelopes.

**Example 12.8**

We apply the Monte Carlo test for departure from CSR to the bodmin data. We obtain the required simulations using the following steps. First we load up the data and obtain $K_{\text{Obs}}(d)$.

```matlab
load bodmin
X = [x,y];
d = 0:.5:10;
nd = length(d);
ux = length(x);
% Now get the Khat for the observed pattern.
khatobs = cskhat(X, bodpoly, d);
```

We are now ready to obtain the $K$-functions for a CSR process through simulation. We use $B = 20$ simulations to obtain the envelopes.

```matlab
% Get the simulations.
B = 20;
% Each row is a Khat from a simulated CSR process.
simul = zeros(B,nd);
for b = 1:B
    [xt,yt] = csbinproc(bodpoly(:,1), bodpoly(:,2), nx);
simul(b,:) = cskhat([xt,yt],bodpoly, d);
end
```

The envelopes are easily obtained using the MATLAB commands `max` and `min`.

```matlab
% Get the envelopes.
khatup = max(simul);
khatlo = min(simul);
% And plot the results.
plot(d,khatobs,'k',d,khatup,'k--',d,khatlo,'k--')
```

In Figure 12.15, we show the upper and lower envelopes along with the estimated $K$-function $K_{\text{Obs}}(d)$. We see from this plot that at the very small scales, there is no evidence for departure from CSR. At some scales there is evidence for clustering and at other scales there is evidence of regularity.

© 2002 by Chapman & Hall/CRC
Example 12.9

In Example 12.6, we estimated the $K$-function for the cardiff data. A plot of the associated $L$-function (see Figure 12.12) showed clustering at those scales. We use the simulation approach to determine whether these results are significant. First we get the estimate of the $L$-function as before.

```matlab
load cardiff
X = [x,y];
d = 0:30;
nd = length(d);
rx = length(x);
khatobs = cskhat(X, cardpoly, d); % Get the \texttt{lhat} function.
lhatobs = sqrt(khatobs/pi) - d;
```

Now we do the same simulations as in the previous example, estimating the $K$-function for each CSR sample. Once we get the $K$-function for the sample, it is easily converted to the $L$-function as shown.

![Figure 12.15](image.png)

In this figure, we have the results of testing for departures from CSR based on $\hat{K}$ using simulation. We show the upper and lower simulation envelopes for the Bodmin Tor data. At small scales (approximately $d < 2$), the process does not show departure from CSR. This is in agreement with the nearest neighbor results of Figure 12.14. At other scales (approximately $2 < d < 6$), we have evidence for clustering. At higher scales (approximately $7.5 < d$), we see evidence for regularity.
% Get the simulations.
B = 20;
% Each row is a Khat from a simulated CSR process.
simul = zeros(B, nd);
for b = 1:B
    [xt, yt] = csbinproc(cardpoly(:, 1), ...
        cardpoly(:, 2), nx);
    temp = cskhat([xt, yt], cardpoly, d);
    simul(b, :) = sqrt(temp / pi) - d;
end

We then get the upper and lower simulation envelopes as before. The plot is shown in Figure 12.16. From this, we see that there seems to be compelling evidence that this is a clustered process.

% Get the envelopes.
lhatup = max(simul);
lhatlo = min(simul);
plot(d, lhatobs, 'k', d, lhatup, 'k--', d, lhatlo, 'k--')

% FIGURE 12.16
The upper and lower envelopes were obtained using 20 simulations from a CSR process. Since the $L$-function lies above the upper envelope, the clustering is significant.
12.5 Simulating Spatial Point Processes

Once one determines that the model for CSR is not correct, then the analyst should check to see what other model is reasonable. This can be done by simulation as shown in the previous section. Instead of simulating from a CSR process, we can simulate from one that exhibits clustering or regularity. We now discuss other models for spatial point processes and how to simulate them. We include methods for simulating a homogeneous Poisson process with specified intensity, a binomial process, a Poisson cluster process, an inhibition process, and a Strauss process. Before continuing, we note that simulation requires specification of all relevant parameters. To check the adequacy of a model by simulation one has to “calibrate” the simulation to the data by estimating the parameters that go into the simulation.

**Homogeneous Poisson Process**

We first provide a method for simulating a homogeneous Poisson process with no conditions imposed on the number of events n. Unconditionally, a homogeneous Poisson process depends on the intensity \( \lambda \). So, in this case, the number of events \( n \) changes in each simulated pattern.

We follow the fanning out procedure given in Ross [1997] to generate such a process for a circular region. This technique can be thought of as fanning out from the origin to a radius \( r \). The successive radii where events are encountered are simulated by using the fact that the additional area one needs to travel to encounter another event is exponentially distributed with rate \( \lambda \). The steps are outlined below.

**PROCEDURE - SIMULATING A POISSON PROCESS**

1. Generate independent exponential variates \( X_1, X_2, \ldots \), with rate \( \lambda \), stopping when

   \[
   N = \min\{ n: X_1 + \ldots + X_n > \pi r^2 \} .
   \]

2. If \( N = 1 \), then stop, because there are no events in the circular region.
3. If \( N > 1 \), then for \( i = 1, \ldots, N-1 \), find

   \[
   R_i = \sqrt{\frac{X_1 + \ldots + X_i}{\pi} }. 
   \]
4. Generate \( N - 1 \) uniform (0,1) variates, \( U_1, \ldots, U_{N-1} \).

5. In polar coordinates, the events are given by \((R, 2\pi U_1)\).

Ross [1997] describes a procedure where the region can be somewhat arbitrary. For example, in Cartesian coordinates, the region would be defined between the \( x \) axis and a nonnegative function \( f(x) \), starting at \( x = 0 \). A rectangular region with the lower left corner at the origin is an example where this can be applied. For details on the algorithm for an arbitrary region, we refer the reader to Ross [1997]. We show in Example 12.10 how to implement the procedure for a circular region.

Example 12.10
In this example, we show how to generate a homogeneous Poisson process for a given \( \lambda \). This is accomplished using the given MATLAB commands.

```matlab
% Set the lambda.
lambda = 2;
r = 5;
tol = 0;
i=1;
% Generate the exponential random variables.
while tol < pi*r^2
    x(i) = exprnd(1/lambda,1,1);
tol = sum(x);
i=i+1;
end
x(end)=[];
N = length(x);
% Get the coordinates for the angles.
th = 2*pi*rand(1,N);
R = zeros(1,N);
% Find the R_i.
for i = 1:N
    R(i) = sqrt(sum(x(1:i))/pi);
end
[Xc,Yc]=pol2cart(th,R);
```

The \( x \) and \( y \) coordinates for the generated locations are contained in \( Xc \) and \( Yc \). The radius of our circular region is 5, and the intensity is \( \lambda = 2 \). The result of our sampling scheme is shown in Figure 12.17. We see that the locations are all within the required radius. To verify the intensity, we can estimate it by dividing the number of points in the sample by the area.

```matlab
% estimate the overall intensity
lamhat = length(Xc)/(pi*r^2);
```
Our estimated intensity is $\hat{\lambda} = 2.05$.

**Binomial Process**

We saw in previous examples that we needed a way to simulate realizations from a CSR process. If we condition on the number of events $n$, then the locations are uniformly and independently distributed over the study region. This type of process is typically called a **binomial process** in the literature [Ripley, 1981]. To distinguish this process from the homogeneous Poisson process, we offer the following:

1. When generating variates from the homogeneous Poisson process, the intensity is specified. Therefore, the number of events in a realization of the process is likely to change for each one generated.
2. When generating variates from a binomial process, the number of events in the region is specified.

To simulate from a binomial process, we first enclose the study region $R$ with a rectangle given by
We can generate the $x$ coordinates for an event location from a uniform distribution over the interval $(x_{\text{min}}, x_{\text{max}})$. Similarly, we generate the $y$ coordinates from a uniform distribution over the interval $(y_{\text{min}}, y_{\text{max}})$. If the event is within the study region $R$, then we keep the location. These steps are outlined in the following procedure and are illustrated in Example 12.11.

**PROCEDURE - SIMULATING A BINOMIAL PROCESS**

2. Obtain a candidate location $s_i$ by generating an $x$ coordinate that is uniformly distributed over $(x_{\text{min}}, x_{\text{max}})$ and a $y$ coordinate that is uniformly distributed over $(y_{\text{min}}, y_{\text{max}})$.
3. If $s_i$ is within the study region $R$, then retain the event.
4. Repeat steps 2 through 3 until there are $n$ events in the sample.

**Example 12.11**

In this example, we show how to simulate a CSR point pattern using the region given with the uganda data set. First we load up the data set and find a rectangular region that bounds $R$. 

```matlab
load uganda
% loads up x, y, ugpoly
xp = ugpoly(:,1);
yp = ugpoly(:,2);
n = length(x);
xg = zeros(n,1);
yg = zeros(n,1);
% Find the maximum and the minimum for a 'box' around % the region. Will generate uniform on this, and throw % out those points that are not inside the region. % Find the bounding box.
minx = min(xp);
maxx = max(xp);
miny = min(yp);
maxy = max(yp);
```

Now we are ready to generate the locations, as follows.

```matlab
% Now get the points.
i = 1;
cx = maxx - minx;
cy = maxy - miny;
while i <= n
```

\begin{equation}
\{(x, y) : x_{\text{min}} \leq x \leq x_{\text{max}}, \ y_{\text{min}} \leq y \leq y_{\text{max}}\}.
\end{equation}
xt = rand(1)*cx + minx;
yt = rand(1)*cy + miny;
k = inpolygon(xt, yt, xp, yp);
if k == 1
    % it is in the region
    xg(i) = xt;
yg(i) = yt;
i = i+1;
end
end

In Figure 12.18, we show a realization of this process. Note that this does look like a CSR process generated these data, unlike the point pattern for the actual crater locations.

![Generated Data Using Binomial Process](image)

FIGURE 12.18
This shows a point pattern generated according to a binomial process.

**Poisson Cluster Process**

We can generate a Poisson cluster process by including a spatial clustering mechanism into the model. First, parent events form a homogeneous Poisson process. Each parent gives rise to a random number of offspring according to some probability distribution $f$. The positions of the children relative to their
parents are independently distributed according to a bivariate distribution \( g \). The events retained in the final pattern are the child events only. The resulting process is isotropic if \( g \) is radially symmetric.

To simulate this type of pattern, we first simulate the parents from a homogeneous Poisson process. Note that the parents should be simulated over a region that is larger than the study region. This is to ensure that edge effects are avoided. Parents outside the study region can have offspring that are in \( R \), so we want to account for those events. For each parent event, we determine the number of offspring by randomly sampling from \( f \). The next step is to locate the number of children around each parent event according to \( g \). The steps for this procedure are outlined here.

**PROCEDURE - SIMULATING A POISSON CLUSTER PROCESS**

1. Simulate the desired number of parents over a region that is slightly larger than the study region \( R \). The parents are generated according to a CSR process.
2. Generate the number of children for each parent according to a probability distribution \( f \). One reasonable choice is to have a Poisson number of children.
3. Generate the locations for each child around the parent according to a bivariate probability distribution \( g \). For example, \( g \) could be multivariate normal, with the mean given by the parent location.
4. Save only the child events that are within the study region.

In the following example, we apply this procedure to generate a Poisson cluster process over the unit square.

**Example 12.12**

We now show how to generate a Poisson cluster process using MATLAB. We first generate 15 parents from a binomial process over a region that is slightly larger.

```matlab
npar = 15;
% Get the vertices for the regions.
rx = [0 1 1 0 0];
ry = [0 0 1 1 0];
rxp = [-.05 1.05 1.05 -.05 -.05];
ryp = [-.05 -.05 1.05 1.05 -.05];
% Get all of the parents.
[xp,yp] = csbinproc(rxp, ryp, npar);
```

We use a Poisson distribution with mean \( \lambda = 15 \) to generate the number of children for the parents.
lam = 15;
% Get the number of children per parent.
nchild = poissrnd(lam,1,npar);

Now we find the locations of the children around the parent using a bivariate normal distribution that is centered at each parent. The covariance of the distribution is given by $\sigma^2 I$, where $I$ is a $2 \times 2$ identity matrix. The value given to the variance $\sigma^2$ would govern the spread of the cluster of children around the parent.

X = [];
sig = r*eye(2);
r = 0.05;
% Locate the children.
for i = 1:npar
    xc = randn(nchild(i),2)*sig + ...
        repmat([xp(i) yp(i)],nchild(i),1);
    X = [X; xc];
end

To get the final events for our sample, we need to determine which ones are inside the study region $R$. We do this using the MATLAB function `inpolygon`. In Figure 12.19, we show the resulting spatial sample. We provide a function called `csclustproc` that will generate patterns that follow a Poisson cluster process.

% Find the ones that are in the region of interest.
ind = find(inpolygon(X(:,1), X(:,2), rx, ry));
% Those are the children for the sample.
x = X(ind,1);
y = X(ind,2);

Inhibition Process

An inhibition process is one that often shows regularity. To simulate this type of process, we include a mechanism in the model that stipulates a minimum distance between two events. We call this distance the inhibition distance $\delta$.

One way to obtain such a process is to first generate a homogeneous Poisson process over the region. The events are then thinned by deleting all pairs of events that are closer than $\delta$. Implementing this procedure in MATLAB is left as an exercise.

Another method is to generate a homogeneous Poisson process one event at a time and discard candidate events if they are within distance $\delta$ of any previously retained event. This type of process is sometimes referred to as Sequential Spatial Inhibition or SSI [Ripley, 1981]. It is important to keep in mind that if the inhibition distance is too large for the region $R$, then it might
be difficult (if not impossible) to generate the required number of points. In Example 12.13, we provide the MATLAB code to generate an inhibition spatial point pattern using this procedure.

**Example 12.13**

To start the procedure, we set the boundary for the region and the inhibition distance.

```matlab
delta = 0.1;
% Get the vertices for the regions.
rx = [0 2 2 0 0];
ry = [0 0 2 2 0];
n = 100;
```

We generate the initial event from a CSR process. Subsequent events are generated and kept if they are not closer than \( \delta \) to any existing events.

```matlab
X = zeros(n,2);
% Generate the first event.
X(1,:) = csbinproc(rx,ry,1);
i = 1;
% Generate the other events.
while i<n
```

![Figure 12.19](image)

*This sample was generated according to a Poisson cluster process.*
[sx, sy] = csbinproc(rx, ry, 1);
x = [sx sy ; X(1:i,:)];

% Find the distance between the events
dist = pdist(x);

% Find the distance between the candidate event
% and the others that have been generated already.
ind = find(dist(1:i) <= delta);

if isempty(ind)
  % Then we keep the event.
  i = i+1;
  X(i,:) = [sx, sy];
endif
end

To verify that no two events are closer than \( \delta \), we find the smallest distance as follows.

% Verify that all are no closer than the
% inhibition distance.
dist = pdist(X);
delhat = min(dist);

For this spatial point pattern, we get a minimum distance of 0.1008. A point pattern generated according to this procedure is shown in Figure 12.20.

FIGURE 12.20
This spatial point pattern was generated under the SSI inhibition process.
The Strauss process [Ripley, 1981] is a point pattern where a specified fraction of events is allowed within a distance $\delta$ of any given event. To generate such a pattern, the first event is located uniformly in $\mathbb{R}$. Other event locations are generated sequentially, similar to the SSI process. If there are existing events within radius $\delta$ of the candidate location, then it is accepted with probability $c^m$, with $m$ representing the number of events closer than $\delta$. The inhibition parameter is given by $c$, which can take on values in the interval $[0, 1]$.

The inhibition parameter specifies the fraction of events allowed within the inhibition distance. If $c = 0$, then the resulting process is the same as SSI. As with the SSI process, care should be taken when specifying the parameters for the process to ensure that the required number of events can be generated. We outline below the steps to generate a spatial point pattern that follows a Strauss process.

**PROCEDURE - SIMULATING A STRAUSS PROCESS**

1. Choose the parameters $n$, $c$, and $\delta$.
2. Generate the first event location $s_1$ uniformly on $\mathbb{R}$ (from a CSR process).
3. Generate a candidate location $s_i$ uniformly on $\mathbb{R}$.
4. If $m = 0$
   
   accept the candidate event $s_i$

   Else if $U \leq c^m$
   
   accept the candidate event $s_i$
5. Repeat steps 3 and 4 until there are $n$ locations in the sample.

It should be noted that we are conditioning on the number of points $n$ in the region. So, in this case, we should consider this a conditional Strauss process.

**Example 12.14**

We now implement the above procedure in MATLAB. We generate a spatial point pattern of size 100 from a Strauss process over a rectangular region. The inhibition distance is $\delta = 0.1$, and the inhibition parameter is $c = 0.5$. We start by setting these parameters and the boundary of the study region.

```matlab
delta = 0.1;
% Get the vertices for the regions.
xr = [0 1 1 0 0];
yr = [0 0 2 2 0];
% Set number of data points.
n = 100;
% Set the inhibition parameter.
```
\begin{verbatim}
c = 0.5;
X = zeros(n,2);
% Generate the first point.
X(1,:) = csbinproc(rx,ry,1);

The following code is similar to the SSI process, except that we now have a mechanism for accepting points that are closer than the inhibition distance.

i = 1;
while i<n
    [sx,sy] = csbinproc(rx, ry, 1);
    xt = [sx sy ; X(1:i,:)];
    % Find the distance between the events.
    dist = pdist(xt);
    % Find the distance between the candidate event and the others that have been generated already.
    ind = find(dist(1:i) <= delta);
    m = length(ind);
    if m == 0
        % Then ok to keep the point - nothing is close.
        i = i+1;
        X(i,:) = [sx, sy];
    elseif rand(1) <= c^m
        % The ok to keep the point.
        i = i+1;
        X(i,:) = [sx, sy];
    end
end

A spatial point pattern generated from these commands is shown in Figure 12.21.
\end{verbatim}

\section*{12.6 MATLAB Code}

The MathWorks has a Mapping Toolbox for MATLAB, which has some functions for spatial statistics. However, the techniques are mostly applicable to geostatistical data. There is also a user-written Spatial Statistics Toolbox that can be downloaded from the internet at

\url{http://www.spatial-statistics.com/}

As with the Mapping Toolbox, this has functions mostly for continuous spatial data.
We provide functions with the Computational Statistics Toolbox that implement most of the techniques that are described in this chapter. These functions are listed in Table 12.1.

**TABLE 12.1**
List of functions from Chapter 12 Included in the Computational Statistics Toolbox

<table>
<thead>
<tr>
<th>Purpose</th>
<th>MATLAB Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>These functions are used to generate samples from various spatial point processes.</td>
<td>csbinproc csclustproc csinhibproc cspoissproc csstraussproc</td>
</tr>
<tr>
<td>This function enables the user to interactively find a study region.</td>
<td>csgetregion</td>
</tr>
<tr>
<td>This is used to estimate the intensity using the quartic kernel. It ignores edge effects.</td>
<td>csintkern</td>
</tr>
<tr>
<td>These functions pertain to the second-order effects of a spatial point pattern.</td>
<td>csfhat csghat cskhat</td>
</tr>
</tbody>
</table>

**FIGURE 12.21**
This spatial point pattern was generated from a Strauss process with $\delta = 0.1$ and $c = 0.5$.
12.7 Further Reading

For information on the theory for all types of spatial data analysis, we highly recommend Cressie [1993] for a comprehensive treatment of the subject. This text is suitable for scientists and engineers at the graduate level. Those areas that require a higher level of mathematics background are clearly marked. The book has many excellent features, among which are lots of examples that illustrate the concepts and the inclusion of spatial data sets.

We already mentioned the text by Bailey and Gatrell [1995]. This book is another excellent resource for spatial statistics. It includes a discussion of the three types of spatial data (point patterns, geostatistical and lattice data), as well as a fourth type dealing with spatial interaction data. The text has many examples and is easy to understand. For a collection of papers on spatial statistics, we refer the reader to Arlinghaus [1996]. This handbook contains many examples of the application of spatial statistics.

For books that focus mainly on spatial point patterns, we refer the reader to Ripley [1981] and Diggle [1983]. Isaaks and Srivastava [1989] and Journel and Huijbregts [1978] are two texts that discuss geostatistical data. For information on the analysis of lattice data, we recommend Cliff and Ord [1981] and Haining [1993].
Exercises

12.1. We mention in the text that there might be an attribute associated with the spatial point pattern. One way to view this attribute would be to plot the value at each event location rather than the plotting symbol. Load the `okblack` data set. Randomly generate some numbers that would correspond to the dollar amount of the theft at each location. Plot these numbers (attributes) at the locations using the `text` command. Keep in mind that you have to convert the numbers to strings before plotting.

12.2. Repeat the procedure in Example 12.4 using bandwidths of $h = 100, 500$. Plot the estimated intensities. How do they differ from the results in Example 12.4? Which bandwidth is better?

12.3. Using the `bodmin` data, plot a dot map. Does it look like a cluster process is a good model for these events?

12.4. Load the `okwhite` data set. Use the `csgetregion` function to interactively select a boundary. Simply click with the left mouse button at the locations of the vertices for the region. There is no need to close the region. When you are done selecting vertices, right click anywhere in the figure window. The output from this function is a set of vertices for the study region. Plot the event locations and the region.

12.5. Explore the Oklahoma City data sets. Estimate the first-order properties and the second-order properties for both patterns. Do the two sets follow different models?

12.6. Write a MATLAB function that will generate an inhibition process using the thinning approach.

12.7. Repeat Example 12.7 for the point-event nearest neighbor distance distribution. Do you arrive at similar conclusions?

12.8. Repeat Example 12.5. Plot the expression given in Equation 12.12 versus $w$. Does this indicate evidence for departure from CSR?

12.9. The test given in Equation 12.22 suffers from two problems: 1) it is for a fixed $x$, and 2) it is not a powerful test. An alternative would be to use the following test statistic

$$T = \max_x \left| F_{\text{obs}}^\wedge(x) - \hat{F}_{\text{CSR}}(x) \right|.$$ 

Use the Monte Carlo techniques of Chapter 6 to determine whether or not there is significant evidence to reject the null hypothesis (that the point process is CSR). What type of departure from CSR would
a large value of $T$ indicate? What type of departure from CSR would a small value of $T$ indicate [Cressie, 1993, p. 636]?

12.10. Generate a realization of a Poisson cluster process. Use your test from problem 12.9 to see if there is significant evidence of clustering.

12.11. Generate a realization of an inhibition process. Apply the nearest-neighbor exploratory graphical techniques ($F$ and $G$ distributions, $K$- and $L$-functions) to see if there is evidence of regularity. Apply the simulation envelope methods to verify that it exhibits regularity.