6

POINT PATTERN ANALYSIS USING R

6.1 INTRODUCTION

In this and the next chapter, some key ideas of spatial statistics will be outlined, together with examples of statistical analysis based on these ideas, via R. The two main areas of spatial statistics that are covered are those relating to *point patterns* (this chapter) and *spatially referenced attributes* (next chapter). One of the characteristics of R, as open source software, is that R packages are contributed by a variety of authors, each using their own individual styles of programming. In particular, for point pattern analysis the `spatstat` package is often used, while for spatially referenced attributed, `spdep` is favoured. One the one hand `spdep` handles spatial data in the same way as `sp`, `maptools` and `GISTools`, while on the other hand `spatstat` does not. Also, for certain specific tasks, other packages may be called upon whose mode of working differs from either of these packages. While this may seem a daunting prospect, the aim of these two chapters is to introduce the key ideas of spatial statistics, as well as providing guidance in the choice of packages, and help in converting data formats. Fortunately, although some packages use different data formats, conversion is generally straightforward, and examples will appear throughout the chapters, whenever necessary.

6.2 WHAT IS SPECIAL ABOUT SPATIAL?

In one sense, the motivations for statistical analysis of spatial data are the same as those for non-spatial data:

- To explore and visualise the data
- To create and calibrate models of the process generating the data
- To test hypotheses related to the processes generating the data

However, a number of these requirements are strongly influenced by the nature of spatial data. The study of mapping and cartography may be regarded as an entire subject area within the discipline of information visualisation, which focuses exclusively on geographical information. In addition, the kinds of hypotheses one might associate with spatial data are quite distinctive – for example, focusing on the detection and location of spatial clusters of events, or on whether two kinds of event (say, two different types of crime) have the same spatial distribution. Similarly, models that are appropriate for spatial data are distinctive, in that they often have to allow for spatial autocorrelation in their random component – for example, a regression model generally includes a random error term, but if the data are spatially referenced, one might expect nearby errors to be correlated. This differs from a 'standard' regression model where each error term is considered to apply independently, regardless of location. In the remainder of this section, point patterns (one of two key types of spatial data considered in this book) will be considered. First, these will be described.

6.2.1 Point Patterns

Point patterns are collections of geographical points assumed to have been generated by a random process. In this case, the focus of inference and modelling is on model(s) of the random processes and their comparison. Typically, a point dataset consists of a set of observed (x, y) coordinates, say $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where n is the number of observations. As an alternative notation, each point could be denoted by a vector $\mathbf{x}_i$, where $\mathbf{x}_i = (x_i, y_i)$. Using the data formats used in `sp`, `maptools` and so on, these data could be represented as `SpatialPoints` or `SpatialPointsDataFrame` objects. Since these data are seen as random, many models are concerned with the probability densities of the random points, $\nu(x_i)$.

Another area of interest is the *interrelation* between the points. One way of thinking about this is to consider the probability density of one point $\mathbf{x}_i$ conditional on the remaining points $\{\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_n\}$. In some situations $\mathbf{x}_i$ is independent of the other points. However, for other processes this is not the case. For example, if $\mathbf{x}_i$ is the location of the reported address for a contagious disease, then it is more likely to occur near one of the points in the dataset (due to the nature of contagion), and therefore not independent of the values of $\{\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_n\}$.

Also important is the idea of a *marked process*. Here, random sets of points drawn from a number of different populations are superimposed (e.g. household burglaries using force and household burglaries not using force) and the relationship between the different sets is considered. The term 'marked' is used here as the dataset can be viewed as a set of points where each point is tagged (or marked) with its parent population. Using the data formats used by `sp`, a marked process could be represented as a spatial points data frame – although the `spatstat` package uses a different format.

6.3 TECHNIQUES FOR POINT PATTERNS USING R

Having outlined the two main data types that will be considered, and the kinds of model that may be applied, in this section more specific techniques will be discussed, with examples of how they may be carried out using R. In this section, we will focus on random point patterns.

6.3.1 Kernel Density Estimates

The simplest way to consider random two-dimensional point patterns is to assume that each random location $\mathbf{x}_i$ is drawn independently from an unknown distribution with probability density function $f(\mathbf{x}_i)$. This function maps a location (represented as a two-dimensional vector) onto a probability density. If we think of locations in space as a very fine pixel grid, and assume a value of probability density is assigned to each pixel, then summing the pixels making up an arbitrary region on the map gives the probability that an event occurs in that area. It is generally more practical to assume an *unknown f*, rather than, say, a Gaussian distribution, since geographical patterns often take on fairly arbitrary shapes – for example, when applying the technique to patterns of public disorder, areas of raised risk will occur in a number of locations around a city, rather than a simplistic radial 'bell curve' centred on the city's mid-point.

A common technique used to estimate $f(\mathbf{x}_i)$ is the *kernel density estimate* (KDE: Silverman, 1986). KDEs operate by averaging a series of small 'bumps' (probability distributions in two dimensions, in fact) centred on each observed point. This is illustrated in Figure 6.1. In algebraic terms, the approximation to $f(\mathbf{x})$, for an arbitrary location $\mathbf{x} = (x, y)$, is given by

$$\hat{f}(\mathbf{x}) = \hat{f}(x,y) = \frac{1}{nh_x h_y}\sum_i k\left(\frac{x-x_i}{h_x}, \frac{y-y_i}{h_y}\right) \tag{6.1}$$

Each of the 'bumps' (central panel in Figure 6.1) map onto the kernel function $k\left(\frac{x-x_i}{h_x}, \frac{y-y_i}{h_y}\right)$ in equation (6.1) and the entire equation describes the 'bump averaging' process, leading to the estimate of probability density in the right-hand panel. Note that there are also parameters h_x and h_y (frequently referred to as the *bandwidths*) in the x and y directions; their dimension is length, and they represent the radii of the bumps in each direction. Varying h_x and h_y alters the shape of the estimated probability density surface – in brief, low values of h_x and h_y lead to very 'spiky' distribution estimates, and very high values, possibly larger than the span of the $\mathbf{x}_i$ locations, tend to 'flatten' the estimate so it appears to resemble the k-function itself; effectively this gives a superposition of nearly identical k-functions with relatively small perturbations in their centre points.

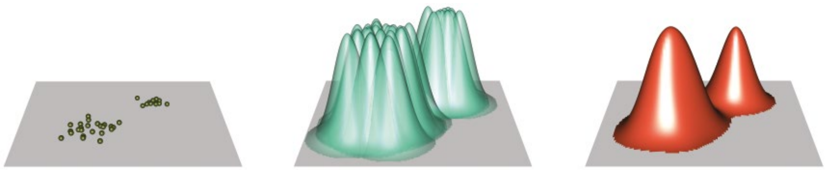

Figure 6.1 Kernel density estimation: initial points (left); bump centred on each point (centre); average of bumps giving estimate of probability density (right)

This effect of varying h_x and h_y is shown in Figure 6.2. Typically h_x and h_y take similar values. If one of these values is very different in magnitude than the other, kernels elongated in either the x or y direction result. Although this may be useful when there are strong directional effects, we will focus on the situation where values are similar for the examples discussed here. To illustrate the results of varying the bandwidths, the same set of points used in Figure 6.1 is used to provide KDEs with three different values of h_x and h_y: on the left, they both take a very low value, giving a large number of peaks; in the centre, there are two peaks; and on the right, only one.

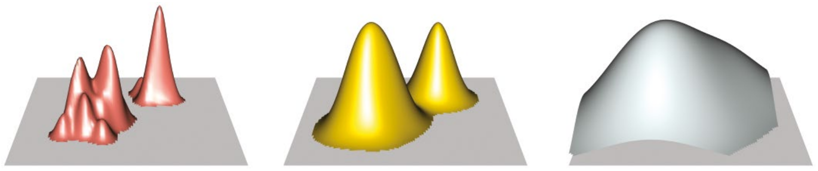

Figure 6.2 Kernel density estimation bandwidths: h_x and h_y too low (left); h_x and h_y appropriate (centre); h_x and h_y too high (right)

An obvious problem is that of choosing appropriate h_x and h_y given a dataset $\{\mathbf{x}_i\}$. There are a number of formulae to provide 'automatic' choices, as well as some more sophisticated algorithms. Here, a simple rule is used, as proposed by Bowman and Azzalini (1997) and Scott (1992):

$$h_x = \sigma_x \left(\frac{2}{3n} \right)^{1/6} \tag{6.2}$$

where σ_x is the standard deviation of the x_i. A similar formula exists for h_y, replacing σ_x with σ_y, the standard deviation of the y_i. The central KDE in Figure 6.2 is based on choosing h_x and h_y using this method.

6.3.2 Kernel Density Estimation Using R

Here, the breaches of the peace (public disturbances) in New Haven, Connecticut are used as an example; recall that this is provided in the `GISTools` package, here loaded using `data(newhaven)`. As an initial inspection of the data, look at the locations of breaches of the peace. These can be viewed on an interactive map using the `tmap` package in `view` mode. The following code loads the New Haven data and `tmap`, sets R in `view` mode and produces a map showing the US Census block boundaries and the locations of breach of the peace, on a backdrop of a *CartoDB* map, provided your computer is linked to the internet. The two layers can be interactively switched on or off, and the backdrop can be changed. Here, we will generally use the default backdrop as it is monochrome, and the information to be mapped will be in colour. The initial map window is seen in Figure 6.3.

```
# Load GISTools (for the data) and tmap (for the mapping)
require(GISTools)
require(tmap)

# Get the data
data(newhaven)
# look at it
# select 'view' mode
tmap_mode('view')
# Create the map of blocks and incidents
tm_shape(blocks) + tm_borders() + tm_shape(breach) +
  tm_dots(col='navyblue')
```

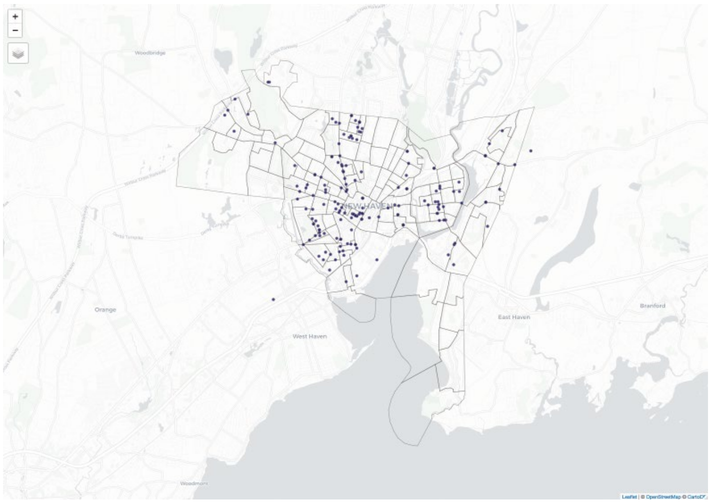

Figure 6.3 Web view mode of **`tmap`**

There are a number of packages in R that provide code for computing KDEs. Here, the `tmap` and `tmaptools` libraries provide some very useful tools. The function to compute kernel density estimation is `map_smooth` from `tmaptools`. This estimates the value of the density over a grid of points, and returns the result as a list – a `raster` object – referred to as `X$raster` (where `X` is the value returned from `map_smooth`), a contour object (`X$iso`) and a polygon object (`X$polygon`). The first of these is a raster grid of values for the KDEs, and the second and third relate to contour lines associated with the KDE; `iso` provides a set of lines (the contour lines) which may be plotted. Similarly, the `polygons` item provides a solid list of polygons that may be plotted (as filled polygons). `map_smooth` takes several arguments (most notably the set of points to use for the KDE) but also a number of optional arguments. Two key ones here are the `bandwidth` and the `cover`. The bandwidth is a vector of length 2 containing h_x and h_y, and the cover is a geographical object whose outline forms the boundary of the locations where the KDE is estimated. Both of these have defaults: the default bandwidth is $\frac{1}{50}$ times the shortest side of the bounding box of the points, and the default cover is the bounding box of the points. However, as discussed earlier, more appropriate h_x and h_y values may be found using (6.2). This is not provided as part of `smooth_map`, but a function is easily written. The division of the result by 1000 is because the projected data are measured in metres, but `smooth_map` expects bandwidths in kilometres.

```
# Function to choose bandwidth according to Bowman and Azzalini / Scott's rule
# for use with smooth_map in tmaptools
```

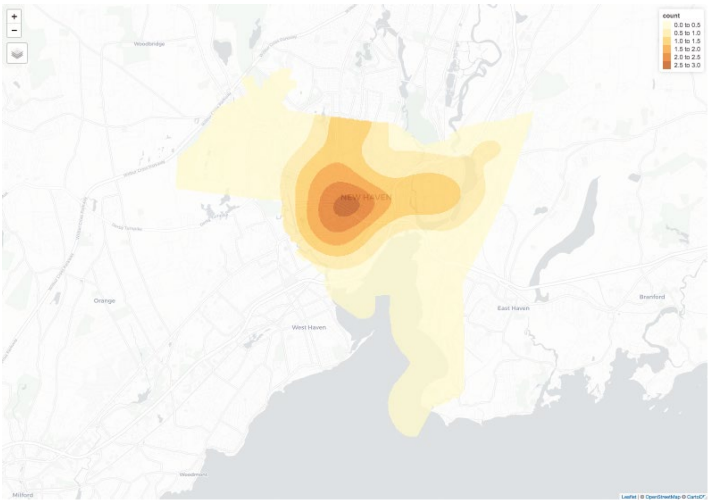

Figure 6.4 KDE map for breaches of the peace

```
choose_bw <- function(spdf) {
  X <- coordinates(spdf)
  sigma <- c(sd(X[,1]),sd(X[,2]))  * (2 / (3 * nrow(X))) ^ (1/6)
  return(sigma/1000)
}
```

Now the code to carry out the KDE and plot the results may be used. Here the `raster` version of the result is used, and plotted on a web mapping backdrop (Figure 6.4).

```
library(tmaptools)
tmap_mode('view')
breach_dens <- smooth_map(breach,cover=blocks, bandwidth = choose_bw(breach))
tm_shape(breach_dens$raster) + tm_raster()
```

The 'count' caption here indicates that the probability densities have been rescaled to represent intensities – by multiplying the KDE by the number of cases. With this scale, the quantity being mapped is the expected number of cases per unit area in the amount of time of the study period.

It is also possible to use the other forms of result (polygons or isolines) to plot the KDE outcomes. In the following code, isolines are produced, again with a backdrop of a web map (see Figure 6.5).

```
tmap_mode('view')
tm_shape(blocks)+ tm_borders(alpha=0.5) +
  tm_shape(breach_dens$iso) + tm_lines(col='darkred',lwd=2)
```

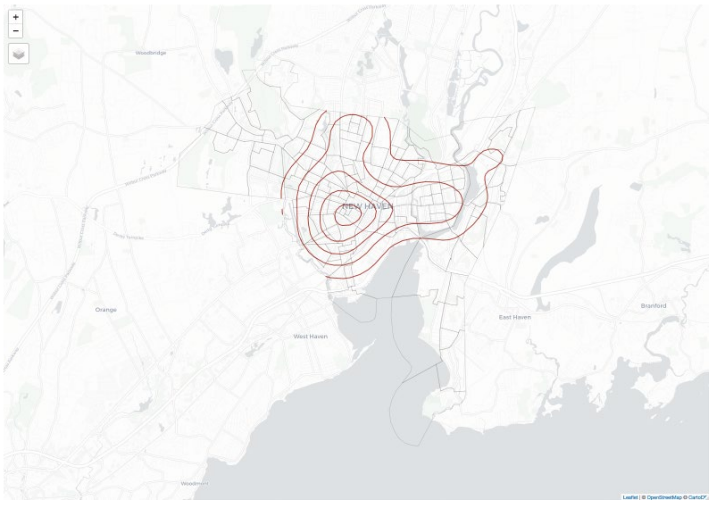

Figure 6.5 KDE map for breaches of the peace – isoline version

Here, a backdrop of block boundaries has also been added to emphasise the limits of the data collection region. In this and the previous map, it is important to be aware of the boundaries of the data sampling region. Low probability densities outside this region are quite likely due to no data being collected there – not necessarily low incident risk!

Self-Test Question 1. As a further exercise, create the *polygons* version of the KDE map in the `plot` mode of `tmap` – the `tm_fill()` function will shade the polygons. As there will be no backdrop map, roads and blocks should be added to the map to provide context. Also, add a map scale.

I

As well as estimating the probability density function *f*(*x*, *y*), kernel density estimation also provides a helpful visual tool for displaying point data. Although plotting point data directly can show all of the information in a small dataset, if the dataset is larger it is hard to discriminate between relative densities of points: essentially, when points are very closely packed, the map symbols begin to overprint and exact numbers are hard to determine; this is illustrated in Figure 6.6. On the left is a plot of locations. The points plotted are drawn from a two-dimensional Gaussian distribution, and their relative density increases towards the centre. However, except for a penumbral region, the intensity of the dot pattern appears to have roughly fixed density. As the KDE estimates relative density, this problem is addressed – as may be seen in the KDE plot in Figure 6.6 (right).

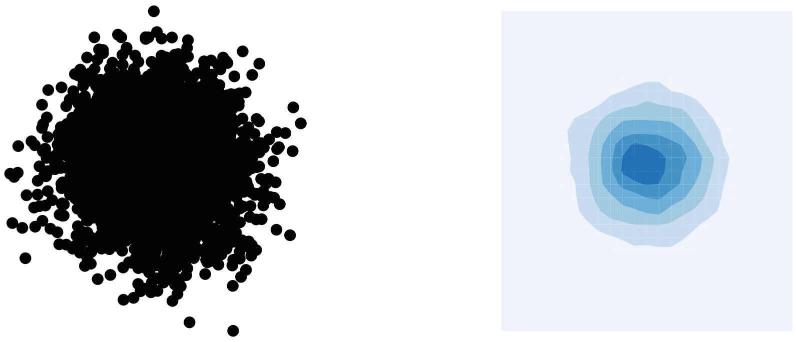

Figure 6.6 The overplotting problem: point plot (left) and KDE plot (right)