7

Correlation

This chapter deals with measuring dependence between points in a point pattern, using the concept of correlation.

7.1 Introduction

Often the motivation for analysing point pattern data is to determine whether the points appear to have been placed independently of each other, or whether they exhibit some kind of interpoint dependence.

Figure 7.1 shows three archetypal point patterns representing 'regularity' (where points tend to avoid each other), 'independence' (complete spatial randomness), and 'clustering' (where points tend to be close together).

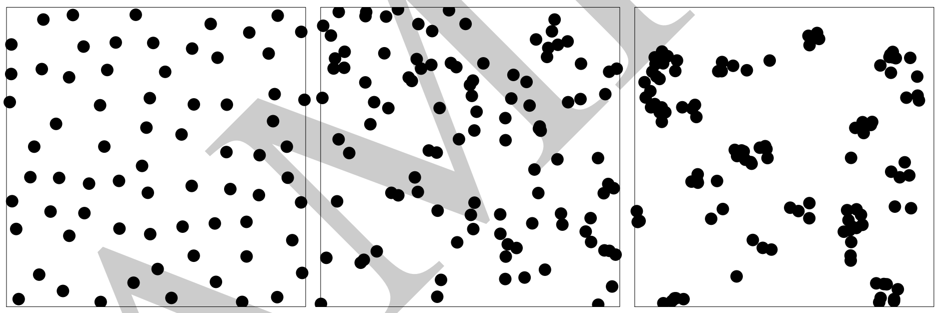

Figure 7.1. *Classical trichotomy between regular (*Left*), independent (*Middle*), and clustered (*Right*) point patterns. All three patterns are in the unit square.*

A standard statistical tool for measuring dependence is *correlation*, or more generally *covariance*. In this chapter we explain how to define and measure covariance in a point process, in such a way that (roughly speaking) the clustered point pattern in the right panel of Figure 7.1 has positive covariance, the completely random pattern in the middle panel has zero covariance, and the regular pattern in the left panel has negative covariance.

In statistical theory, correlation is classified as a *second moment* quantity. The 'first moment' of a random variable X is its mean value; the 'second moment' or 'mean square' is the mean of X^2. Together the first and second moments of random variables determine important quantities such as variance, standard deviation, covariance, and correlation.

Second moment quantities for point processes are intimately related to counting *pairs* of points, or adding up contributions from each pair of points in the process. In a point process $\mathbf{X}$, the squared point count $n(\mathbf{X} \cap B)^2$ can be interpreted as the number of *pairs of points* x_i, x_j in $\mathbf{X}$ which fall in the nominated set B, including identical pairs where $i = j$. The second moment of $n(\mathbf{X} \cap B)$ is the expected number of pairs of points falling in B.

Correlation has the great virtue of being easy to calculate and handle, and is a powerful tool for the data analyst. There are two important caveats. First, accurate measurement of correlation requires faithful estimation of the mean (first moment) if we are to avoid problems of spurious correlation and confounding. In the point process context, this means that we must have good knowledge of the intensity before we can trust the correlation. Second, the correlation is merely a summary index of statistical association, not a characterisation of dependence or causality: 'correlation is not causation'. Using only correlations, we cannot discriminate between different possible causes of spatial clustering.

7.2 Manual methods

We start by describing two simple, manual techniques which motivate and give insight into the more advanced methods for assessing corrlation in point pattern data.

7.2.1 Greig-Smith plot and Morisita index

If the observation window is a rectangle, a simple strategy for assessing spatial correlation is to subdivide the window into rectangular quadrats of equal size, and to count how often a pair of data points falls in the same quadrat. If there are n data points altogether, there are $n(n-1)$ ordered pairs of distinct points. If we use m quadrats and these are found to contain $n_1, \ldots, n_m$ points, respectively, then the jth quadrat contains $n_j(n_j-1)$ ordered pairs of distinct points. The total number of ordered pairs of distinct points which fall inside the same quadrat is thus $\sum_j n_j(n_j-1)$. The ratio

$$\frac{\sum_j n_j(n_j-1)}{n(n-1)}$$

is the fraction of all pairs of data points in which both points fall in the same quadrat. In a completely random (homogeneous Poisson) process, where points are independent of each other, two points fall in the same quadrat with probability $1/m$, where m is the number of quadrats, so the fraction above is expected to equal $1/m$. The ratio of the observed and expected fractions is the *Morisita index* [488]

$$M = m\,\frac{\sum_j n_j(n_j-1)}{n(n-1)}. \tag{7.1}$$

This index should be close to 1 if the points are independent, greater than 1 if they are clustered, and less than 1 if they are regular.

Repeating this calculation using different subdivisions of the window into quadrats, and plotting the Morisita index against the diameter of the quadrats, yields a *Morisita index plot*. Figure 7.2 shows that this has the ability to distinguish between the three archetypal point patterns in Figure 7.1. In `spatstat` the Morisita index plot of a point pattern `X` is generated by `miplot(X)`.

The Morisita index is closely related to the *index of dispersion* for quadrat counts. Suppose we calculate the sample variance of the quadrat counts,

$$s^2 = \frac{1}{m-1}\sum_{j=1}^{m}(n_j-\overline{n})^2$$

where $\overline{n}$ is the average quadrat count, $\overline{n} = n/m$. If the point process is Poisson, then the counts $n_1, \ldots, n_m$ are observations of independent Poisson random variables with the same, unknown mean μ. The variance of a Poisson random variable is equal to its mean. A standard index of overdispersion or underdispersion for count variables is the sample variance divided by the sample mean: for

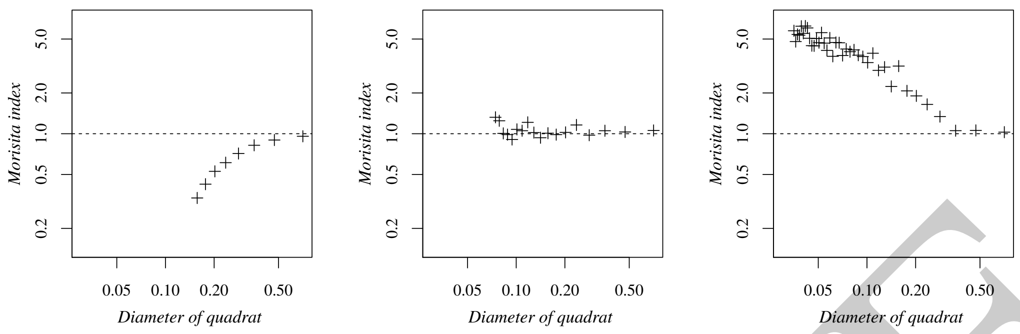

Figure 7.2. *Morisita index plots (on logarithmic scale) for the three patterns in Figure 7.1.*

a Poisson distribution this ratio should be approximately equal to 1. Dividing the sample variance s^2 by the sample mean $\overline{n}$ gives the *index of dispersion*

$$I = \frac{s^2}{\overline{n}} = \frac{m}{n(m-1)} \sum_{j=1}^{m} (n_j - \frac{n}{m})^2.$$

An earlier paper by Greig-Smith [296] had proposed plotting the index of dispersion against quadrat size. A little algebra shows that these are equivalent: the dispersion index I and Morisita index M are directly related by $I = (m/(m-1))[(n-1)M - (n/m-1)]$. As discussed on page 167, the index of dispersion is also closely related to the χ^2 test of CSR based on quadrat counts. Thus, the Morisita and Greig-Smith plots are effectively plots of the test statistic for a χ^2 quadrat counting test of CSR plotted against the size of the quadrats. See [462, 575, 225].

For the purposes of this chapter we will focus on the Morisita index M. Note especially that M *assumes the intensity is homogeneous*. If this is not the case, large values of M could arise simply from spatial inhomogeneity, rather than from some form of correlation between the points.

The Morisita index is not sensitive to subtle differences in spatial scale, because it is based on subdividing the observation window coarsely into quadrats. This suggests that we look for better ways to summarise the information about pairs of points.

There is also something unsatisfactory about the theoretical derivation of the Morisita index given above. That derivation referred only to the homogeneous Poisson point process, and calculated that the Morisita index should be about 1 for that process. But if the data were generated by another kind of point process, it is unclear (at least from the previous discussion) how to interpret the value of the Morisita index. Indeed the Morisita index might not even be a well-defined property of the point process: for example, it might depend on the size of the observation window. Although these questions can be resolved, the answers are not very simple.

A good statistical index should not only be accessible by simple direct calculation from the data, but it should have a simple, direct interpretation for the point process which generated the data. In the rest of the chapter we look for such indices.

7.2.2 Fry plot

More information about spacings in the point pattern can be obtained using a *Fry plot* or Patterson plot. Originally developed for crystallography by Patterson [534, 535] this technique was independently reinvented in geophysics by Fry [271, 321].

A Fry plot can be drawn by hand, as follows (see Figure 7.3). First print the point pattern on a sheet of paper. Take a transparency or sheet of tracing paper, and mark a cross in the middle. Place the transparency over the printout, so that the cross on the transparency lies on one of the data points.

Copy the positions of all the other data points onto the transparency in the same relative position. (Of course some data points may be too far away to be copied onto the transparency, depending on its size.) Now move the cross to another data point, and repeat the copying process. After every data point has been visited, the pattern on the transparency is the Fry plot.

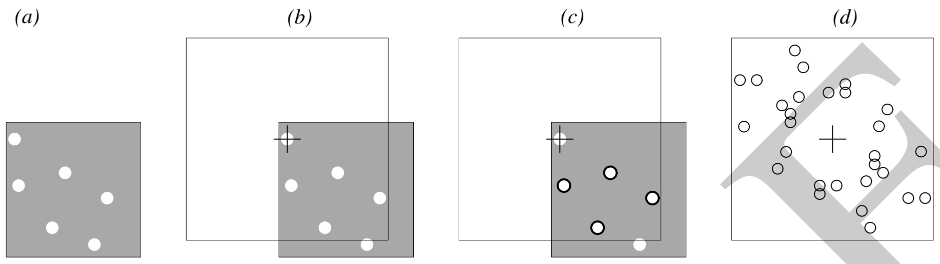

Figure 7.3. *Stages in forming the Fry plot.* (a): *data point pattern, printed on paper.* (b): *transparency superimposed on point pattern so that centre of transparency (+) lies above the first data point.* (c): *other data points are copied (○) onto the transparency.* (d): *final result.*

In mathematical terms the Fry plot is a scatterplot of the vector differences $x_j - x_i$ between all pairs of distinct points in the pattern.

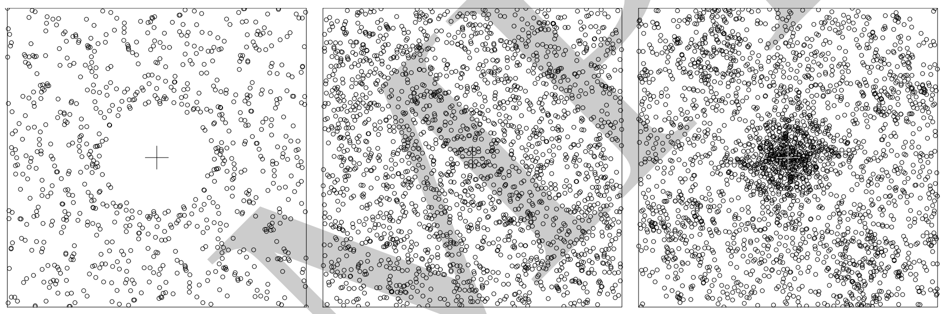

Figure 7.4. *Fry plots for the three patterns in Figure 7.1.*

In `spatstat` the Fry plot of a point pattern `X` is generated by the command `fryplot(X)` or `plot(frypoints(X))`. Additional arguments allow the plot to be restricted to a smaller area around the origin $(0,0)$, where the interesting detail is found, or restricted to certain groups of data points.

Figure 7.4 shows the Fry plots for the three archetypal point patterns in Figure 7.1, restricted to squares of width 0.6 units. The origin is in the middle of each panel of Figure 7.4 and is indicated by the '+' symbol, visible only in the left panel.

The origin in the Fry plot represents a typical point of the point pattern, and the dots in the Fry plot represent the positions of other nearby points, relative to the typical point. In the left panel of Figure 7.4 there is a clear absence of dots in the middle of the panel, indicating that data points never come closer to each other than a certain minimum distance. The middle panel of Figure 7.4 shows no obvious pattern, while the right panel shows a higher concentration of dots near the origin, indicating a clustered pattern. Thus, the Fry plot is easily able to distinguish the three basic kinds of dependence between points.

Fry plots are implicitly based on the assumption that the underlying process is stationary (Section 5.6.3). Under this assumption, the Fry plot contains essentially all information about correlations in the point process.

Fry plots can be very useful for spotting features of the point pattern which might not otherwise be obvious, such as discretisation of the coordinates. In geophysics, Fry plots have proven to be very useful for inferring mechanical strain in rocks, which is reflected in the shape of an elliptical 'hole' in the Fry plot. However, in many other applications, the interpretation of the Fry plot is too subjective. Fry plots often need to be simplified, reduced, or summarized in order to extract usable information.

7.3 The K-function

A very popular technique for analysing spatial correlation in point patterns is the K-function proposed[1] by Ripley [572].

7.3.1 The empirical K-function

Suppose that the primary research question concerns the distance or spacing between points in the point pattern. It would then be natural to measure the distances $d_{ij} = \|x_i - x_j\|$ between all ordered pairs of distinct points x_i and x_j in the point pattern $\mathbf{x}$ under consideration. These distances clearly capture a great deal of information about the spatial pattern. If the pattern is clustered, many of the pairwise distances will be small; if the pattern is regular, few of the distances will be small. This suggests that we might look at a statistical summary of the distances d_{ij}, such as the histogram.

We have argued that a good statistical summary of a point pattern should have a simple, direct interpretation in terms of the *point process* which generated the data. The histogram of observed pairwise distances d_{ij} is difficult to interpret in this way, because it depends on the shape and size of the observation window: the same point process, viewed through different windows, yields different histograms of pairwise distances.

Consider instead the empirical cumulative distribution function of the pairwise distances,

$$\begin{aligned}\widehat{H}(r) &= \text{fraction of values } d_{ij} \text{ less than } r \\ &= \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \mathbf{1}\left\{d_{ij} \leq r\right\} \end{aligned} \tag{7.2}$$

defined for each distance value $r \geq 0$. Here we use the 'indicator' notation: $\mathbf{1}\{\dots\}$ equals 1 if the statement '…' is true, and 0 if the statement is false. The sum is taken over all ordered pairs i, j of indices which are not equal. The sum of these indicators is simply the number of times that the statement is true, that is, the number of values d_{ij} which are less than or equal to r. The denominator $n(n-1)$ is the total number of pairs of distinct points, so $\widehat{H}(r)$ is the fraction of pairs for which the distance is less than or equal to r.

Notice that $\widehat{H}(r)$ is analogous to the Morisita index: both quantities report the fraction of pairs of points which lie close together. The distance argument r in $\widehat{H}(r)$ defines 'closeness', and is analogous to the size of quadrats in the Morisita index.

We can also visualise the calculation of $\widehat{H}(r)$ using the Fry plot. The dots in the Fry plot are the vector differences $x_i - x_j$ between all pairs of distinct points in the point pattern dataset. The lengths of these vectors are the distances d_{ij}. To count the number of distances d_{ij} that are less than or equal to r, we simply draw a circle of radius r, centred at the origin of the Fry plot, and count the number of dots in the Fry plot which fall inside this circle. That is, $\widehat{H}(r)$ is the fraction of dots in the Fry plot which fall inside the circle of radius r.

[1]There are similar concepts in statistical physics [531] and astronomy (cf. [445]).

The contribution from each data point x_i to the sum in (7.2) is

$$t_i(r) = \sum_{j \neq i} \mathbf{1}\left\{d_{ij} \leq r\right\},$$

the number of *other* data points x_j which lie closer than a distance r. We might call this the number of *r-neighbours* for the point x_i. Equivalently $t_i(r)$ is the number of data points which fall inside a circle of radius r centred at x_i, not counting x_i itself. Then

$$\widehat{H}(r) = \frac{1}{n(n-1)} \sum_{i=1}^{n} t_i(r) = \frac{1}{n-1}\, \bar{t}(r)$$

where $\bar{t}(r) = (1/n)\sum_i t_i(r)$ is the average number of r-neighbours per data point.

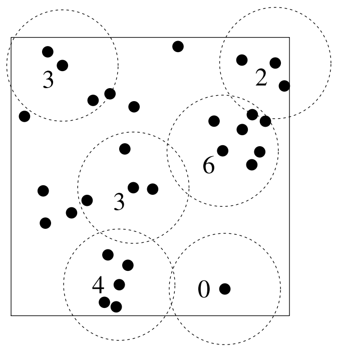

Figure 7.5. *Counting r-neighbours. In each circle, the numeral shows the value of $t_i(r)$ for the data point x_i at the centre of the circle.*

This is an important clue that the quantity we really want to estimate is *the average number of r-neighbours of a typical random point.* Figure 7.5 shows an intermediate stage in the calculation of $\bar{t}(r)$. The number of r-neighbours is shown for each of several data points. The average of these numbers for all data points is $\bar{t}(r)$.

The average number of r-neighbours of a data point will depend on the overall average density of points in the dataset. In a completely random pattern, we would expect $\bar{t}(r)$ to be about $\lambda \pi r^2$, since $\bar{t}(r)$ counts the number of points falling in a circle of radius r which has area πr^2. In order to be able to compare datasets with different numbers of points, it makes sense to standardise $\bar{t}(r)$ by dividing by λ. Since the maximum number of neighbours of any data point is $n-1$, it may be more appropriate to divide by $\tilde{\lambda} = (n-1)/|W|$, where n is the number of points and $|W|$ is the area of the observation window. The result of this standardisation is $\bar{t}(r)/((n-1)/|W|) = |W|\widehat{H}(r)$.

The function $|W|\widehat{H}(r)$ is the standardised average number of r-neighbours of a typical data point. In order to be fully able to compare datasets observed in different windows, we also need to take account of *edge effects* as explained in Section 7.4. This leads to a slight modification of the function $|W|\widehat{H}(r)$, called the **empirical K-function**,

$$\widehat{K}(r) = \frac{|W|}{n(n-1)} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \mathbf{1}\left\{d_{ij} \leq r\right\} e_{ij}(r) \tag{7.3}$$

where $e_{ij}(r)$ is an *edge correction weight* described in Section 7.4.

In summary, the empirical K-function $\widehat{K}(r)$ is the cumulative average number of data points lying within a distance r of a typical data point, corrected for edge effects, and standardised by dividing by the intensity. The standardisation and edge correction make it possible to compare point patterns with different numbers of points, observed in different windows.

Figure 7.6 displays the empirical K-functions for the three archetypal point patterns of Figure 7.1, showing that the empirical K-function clearly has the ability to discriminate between the three basic kinds of interpoint dependence. The empirical K-function for the clustered pattern lies above the empirical K-function for a completely random pattern, which in turn lies above the empirical K-function for a regular pattern. This is equivalent to saying that, after adjusting for intensity, a typical point in the clustered pattern has more close neighbours than a typical point in the completely random pattern, which in turn has more close neighbours than a typical point in the regular pattern.

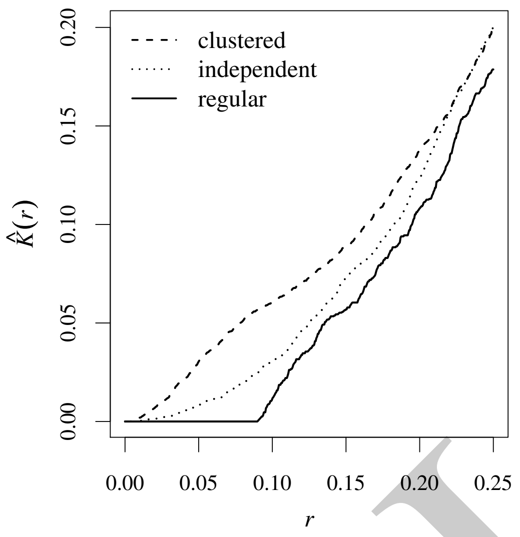

Figure 7.6. *Empirical K-functions for the three patterns in Figure 7.1. Solid line: regular pattern. Dotted line: independent pattern. Dashed line: clustered pattern.*

When interpreting graphs of the empirical K-function, it helps to remember that $\widehat{K}(r)$ is a standardised or relative quantity, rather than a direct physical quantity. The empirical K-function of a completely mapped forest gives us, for each r, the average number of neighbour trees lying within distance r of a typical tree, *divided by* the density of the forest in trees per unit area. The values of $\widehat{K}(r)$ are expressed in units of `(number)/(number/area) = area`. Standardisation makes it possible to compare the degree of regularity or clustering in forests with different average densities of trees. To recover the physically meaningful average number of neighbours $\bar{t}(r)$, we would need to know the forest density $\overline{\lambda}$, and multiply $\widehat{K}(r)$ by $\overline{\lambda}$ to obtain $\bar{t}(r)$.

Warning: using the K-function implicitly assumes that the point process has homogeneous intensity. See Section 7.3.5.

7.3.2 The true K-function of a point process

The empirical function $\widehat{K}(r)$ is a summary of the pairwise distances in the point pattern dataset, normalised to enable us to compare different datasets. But the key question about any summary statistic for a point pattern is what it means for the *point process* which generated the pattern.

The K-function of a point process $\mathbf{X}$ will be defined as the expected number of r-neighbours of a typical point of $\mathbf{X}$, divided by the intensity λ. For this we need to assume[2] that $\mathbf{X}$ is **stationary** (see Section 5.6.3: the distribution of $\mathbf{X}$ is the same as the distribution of the shifted process $\mathbf{X}+v$, for any vector v). This implies that $\mathbf{X}$ has homogeneous intensity λ. We may then define

$$K(r) = \frac{1}{\lambda}\,\mathbb{E}\left[\text{number of } r\text{-neighbours of u} \;\middle|\; \mathbf{X} \text{ has a point at location } u\right] \tag{7.4}$$

for any $r \geq 0$ and any location u. Since the process is stationary, this definition does not depend on the location u. On the right-hand side of (7.4), the symbol '$|$' indicates that this is a conditional expectation. Intuitively, we assume there is a random point of $\mathbf{X}$ at the location u; given this, we

[2]Slightly weaker assumptions are enough, and these will be stated below.

find the expected number of other points of $\mathbf{X}$ lying within a distance r; and finally we divide by the intensity λ, to obtain $K(r)$.

Extending the notation $t_i(r)$, let us define for any spatial location u

$$t(u,r,\mathbf{x}) = \sum_{j=1}^{n(\mathbf{x})} \mathbf{1}\left\{0 < \|u - x_j\| \le r\right\}, \tag{7.5}$$

the number of points in the point pattern $\mathbf{x}$ that lie within a distance r of the location u, but not at u itself.

Definition 7.1. *If* $\mathbf{X}$ *is a stationary point process, with intensity* $\lambda > 0$*, then for any* $r \ge 0$

$$K(r) = \frac{1}{\lambda}\mathbb{E}\left[t(u,r,\mathbf{X}) \;\middle|\; u \in \mathbf{X}\right] \tag{7.6}$$

does not depend on the location u, and is called the K-function *of* $\mathbf{X}$.

Explicit formulae for the K-function have been derived for a few point process models. For the homogeneous Poisson point process (CSR), since the points are independent, intuitively speaking, the presence of a random point at the location u will have no bearing on the presence of points at other locations, so

$$\mathbb{E}\left[t(u,r,\mathbf{X}) \;\middle|\; u \in \mathbf{X}\right] = \mathbb{E}\left[t(u,r,\mathbf{X})\right].$$

But $t(u,r,\mathbf{X})$ is the number of points of $\mathbf{X}$ falling in the disc $b(u,r)$ of radius r centred at u. The expected number of such points is $\lambda \times |b(u,r)| = \lambda\pi r^2$. Dividing by λ shows that, for a homogeneous Poisson process,

$$K_{pois}(r) = \pi r^2 \tag{7.7}$$

regardless of the intensity. This calculation is for two dimensions; for the case of three dimensions see Chapter 15.

7.3.3 Use of the empirical K-function

Visual inspection of empirical K-function

To study correlation in a point pattern dataset, assuming the intensity is homogeneous, we can plot the empirical K-function $\widehat{K}(r)$ calculated from the data, together with the theoretical K-function of the homogeneous Poisson process $K_{pois}(r) = \pi r^2$, which serves as the benchmark of 'no correlation'. Figure 7.7 shows this graphic for each of the three archetypal patterns in Figure 7.1.

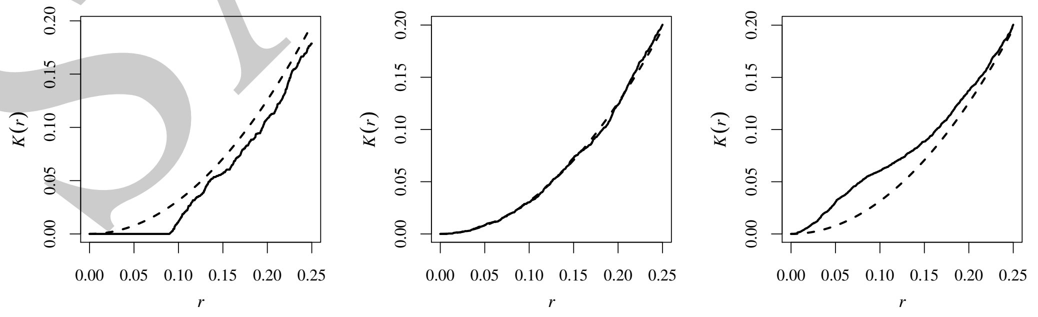

Figure 7.7. *Empirical K-function (solid lines) for each of the three patterns in Figure 7.1, and the theoretical K-function for a Poisson process (dashed lines).*

Figure 7.7 shows that this graphic easily detects the presence and type of correlation between points in a point pattern. In the left panel, the curve for the empirical K-function (solid lines) is lower than the theoretical curve for a completely random pattern (dashed lines), $\widehat{K}(r) < K_{pois}(r)$, indicating that a typical point in this pattern has fewer neighbours than would be expected if the pattern were completely random. This is consistent with a regular point process. Similarly in the right panel, the empirical curve is higher than the theoretical curve, $\widehat{K}(r) > K_{pois}(r)$, indicating that a typical point has more neighbours than would be expected if the pattern were completely random; this is consistent with clustering. The K-function is a powerful tool for investigating point patterns, but we need to remember that 'correlation is not causation'. If analysis shows that $\widehat{K}(r) > K_{pois}(r)$, the careful scientist will not say that this 'indicates' clustering, but that it is 'consistent with' clustering, or that it indicates 'positive association' between points. See Section 7.3.5 for more discussion.

Transformation of K

The *centred* version of the K-function is $K(r) - K_{pois}(r) = K(r) - \pi r^2$. This can be useful for classifying a point pattern as random, clustered, or regular, because the function is zero if the point pattern is completely random. It is less useful for other purposes.

A commonly used transformation of K proposed by Besag [103] is the ***L*-function**

$$L(r) = \sqrt{\frac{K(r)}{\pi}} \tag{7.8}$$

which transforms the theoretical Poisson K-function $K_{pois}(r) = \pi r^2$ to the straight line $L_{pois}(r) = r$, making visual assessment of the graph much easier. Figure 7.8 shows the empirical L-functions for the three archetypal point patterns. The square root transformation also approximately stabilises the variance of the estimator (that is, the variance of the empirical function $\widehat{L}(r)$ is roughly constant as a function of r), making it easier to assess deviations.

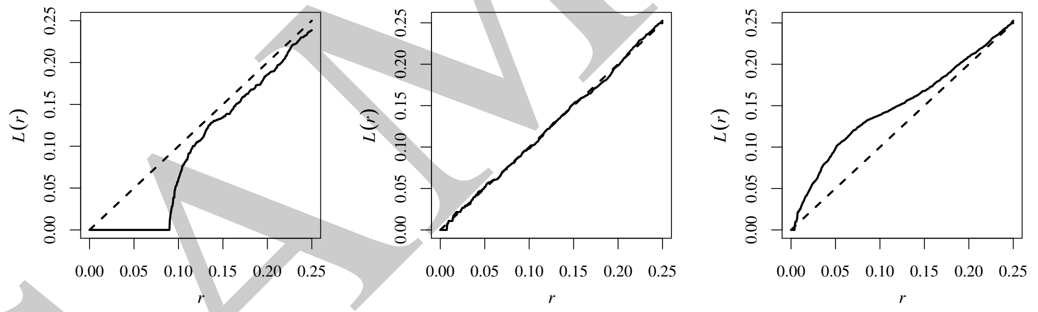

Figure 7.8. *Empirical L-function (solid lines) for each of the three patterns in Figure 7.1, and the theoretical L-function for a Poisson process (dashed lines).*

Besag's L-function has become such a popular transformation of Ripley's K-function that many writers in applied fields, and some software packages, apply the transformation without mentioning it. They plot $L(r)$ against r, or the centred version $L(r) - r$ against r, but still call it Ripley's K-function. This is not advisable because K and L are not equivalent in some contexts. When we say $K(r)$ we shall always mean $K(r)$.

Statistical inference

Options for formal statistical inference about the K-function include *confidence intervals* and *hypothesis tests* (the latter associated with *simulation envelopes*). These ideas are illustrated in Figure 7.9 using the clustered pattern in the right panel of Figure 7.1.

The left panel of Figure 7.9. shows a *confidence interval* for the true K-function of the point pattern. The right panel shows an *acceptance interval* (or 'non-rejection' interval) for testing whether the pattern is completely random.

It is very important to understand the difference between these two techniques. A **confidence interval** is designed to contain the true value of the target quantity with a specified degree of confidence. It is centred around an estimated value of the target quantity, and its width is an indication of the precision of the estimation. In the left panel of Figure 7.9, the shaded region is a pointwise 95% confidence interval for the true K-function of the clustered pattern; the shading is centred around the estimated K-function; the width of the shaded region reflects the precision of the estimate. Section 7.7 explains how to construct confidence intervals for the K-function.

On the other hand, an **acceptance interval** (or 'non-rejection interval') contains the *hypothesised* value of the target quantity. It is the range of values that are deemed to be not significantly different from the hypothesised value. The width of the acceptance interval reflects the inherent variability of the observations when the hypothesis is true. In the right panel of Figure 7.9, the shaded region is the acceptance interval for a formal test, with significance level 0.05, of the hypothesis that the pattern is completely random. The acceptance interval is centred around the K-function of a completely random point pattern, $K_{pois}(r) = \pi r^2$, and the width of the interval reflects the variability of $\widehat{K}(r)$. For a fixed value of r, the test rejects the null hypothesis of CSR if the value of $\widehat{K}(r)$ lies outside this acceptance interval. Section 7.8 explains how to construct acceptance intervals for testing CSR. More explanation and important caveats about hypothesis tests are given in Chapter 10.

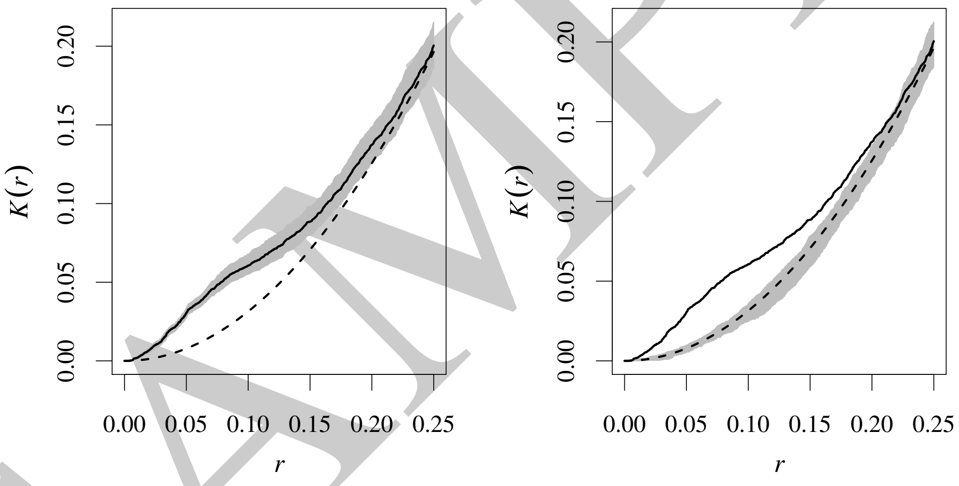

Figure 7.9. *Tools for formal statistical inference, demonstrated on the clustered pattern in the right panel of Figure 7.1.* Left: *95% confidence intervals (shaded) for the true value of the K-function, obtained using Loh's bootstrap (function* `lohboot`*, Section 7.7.3).* Right: *acceptance region (shaded) for a hypothesis test of complete spatial randomness, with significance level 5%, using the envelopes of the K-functions of 39 simulated realisations of CSR. Generated using the function* `envelope` *(Sections 7.8 and 10.7).*

The K-function as a guide for building models

The K-function can serve as a guide for building a point process model of the phenomenon being studied. Chapter 12 presents one class of models which can be fitted directly to point pattern data using the K-function. More commonly the K-function is only a rough guide to the required behaviour of the model.

A point process model of a natural phenomenon could include several stages in which points are randomly added, combined, displaced, or removed. In this regard there are two important properties of the K-function.

Firstly **the K-function is invariant under random thinning**. Suppose $\mathbf{X}$ is a stationary point process, and $\mathbf{Y}$ is the point process obtained by independent random thinning — randomly deleting or retaining each point of $\mathbf{X}$, with a constant probability p of retaining each point (Section 5.3.3). Then the K-function of $\mathbf{Y}$ is identical to the K-function of $\mathbf{X}$.

Secondly there is the effect of **superposition**. Suppose $\mathbf{X}$ and $\mathbf{Y}$ are two independent, stationary point processes, with intensities $\lambda_{\mathbf{X}}, \lambda_{\mathbf{Y}}$ and K-functions $K_{\mathbf{X}}(r), K_{\mathbf{Y}}(r)$. Superimposing these two processes gives a stationary point process with intensity $\lambda = \lambda_{\mathbf{X}} + \lambda_{\mathbf{Y}}$ and K-function

$$K(r) = p_{\mathbf{X}}^2 K_{\mathbf{X}}(r) + p_{\mathbf{Y}}^2 K_{\mathbf{Y}}(r) + 2 p_{\mathbf{X}} p_{\mathbf{Y}} \pi r^2 \tag{7.9}$$

where $p_{\mathbf{X}} = \lambda_{\mathbf{X}}/\lambda$ and $p_{\mathbf{Y}} = \lambda_{\mathbf{Y}}/\lambda$ are the relative proportions of points coming from $\mathbf{X}$ and $\mathbf{Y}$. The *centred* K-functions are more simply related:

$$(K(r) - \pi r^2) = p_{\mathbf{X}}^2 (K_{\mathbf{X}}(r) - \pi r^2) + p_{\mathbf{Y}}^2 (K_{\mathbf{Y}}(r) - \pi r^2). \tag{7.10}$$

7.3.4 Estimating the K- and L-functions in `spatstat`

The `spatstat` function `Kest` computes several estimates of $K(r)$ using different edge corrections. It also returns the ‘benchmark’ value πr^2 which is the theoretical value for $K(r)$ for a homogeneous Poisson process.

```
 > K <- Kest(cells)
 > Ki <- Kest(cells, correction="isotropic")
```

Here `cells` is Ripley’s [572] biological cells dataset (shown on page 93) which we shall use frequently as an example.

The argument `correction` specifies which estimate or estimates will be computed by `Kest`. Options include `"isotropic"` for Ripley’s isotropic correction, `"translation"` for the translation correction, `"rigid"` for the rigid motion correction, `"border"` for the border correction, and `"none"` for the uncorrected estimate. These are explained in Section 7.4. Any number of edge corrections may be selected. Specifying a single edge correction will reduce computation time, especially in simulation experiments.

Additionally `Kest`, like all standard summary functions in `spatstat`, recognises the options `"best"` (representing the estimate with the best statistical performance regardless of computational cost) and `"good"` (the estimate with the best statistical performance for reasonable computational cost).

If no `correction` argument is given, the default is to compute the isotropic, translation, and border corrections, unless the point pattern contains more than `nlarge` points, when only the border correction will be computed. The default threshold is `nlarge = 3000` points. Setting `nlarge=Inf` will suppress this behaviour.

The `spatstat` function `Lest` computes estimates of $L(r)$ directly from point pattern data.

```
 > Lc <- Lest(cells)
```

The arguments of `Lest` are identical to those of `Kest`.

See Section 7.5 for a detailed explanation of how to plot and manipulate estimates of the K-function and other summary statistics. A previously computed K-function can be converted to the L-function using the syntax `L <- with(K, sqrt(./pi))`, explained on page 224, or plotted as an L-function using the syntax `plot(K, sqrt(./pi)~r)`, explained on page 221.

7.3.5 Caveats about the K-function

The use of the K-function for analysing point patterns has become established across wide areas of applied science, following Ripley's influential paper [572] and many subsequent textbooks [190, 221, 225, 661, 575, 576, 638]. Useful and powerful as this methodology is, there is an unfortunate tendency to apply it uncritically and to neglect other methods of analysis.

7.3.5.1 K-function assumes homogeneity

The K-function is defined and estimated under the *assumption that the point process is stationary.* If the process is not stationary, deviations between the empirical and theoretical functions (e.g., $\widehat{K}$ and K_{pois}) are not necessarily evidence of interpoint interaction, since they may also be attributable to variations in intensity [83].

Figure 7.10 shows a realisation of an inhomogeneous Poisson process, and its empirical K-function. The plot of the K-function shows that, on average, a point in this pattern has more r-neighbours than would be expected for CSR. However, this happens because there is a higher density of points in one corner of the window. The points are positively associated, but not because they are clustered in the usual sense.

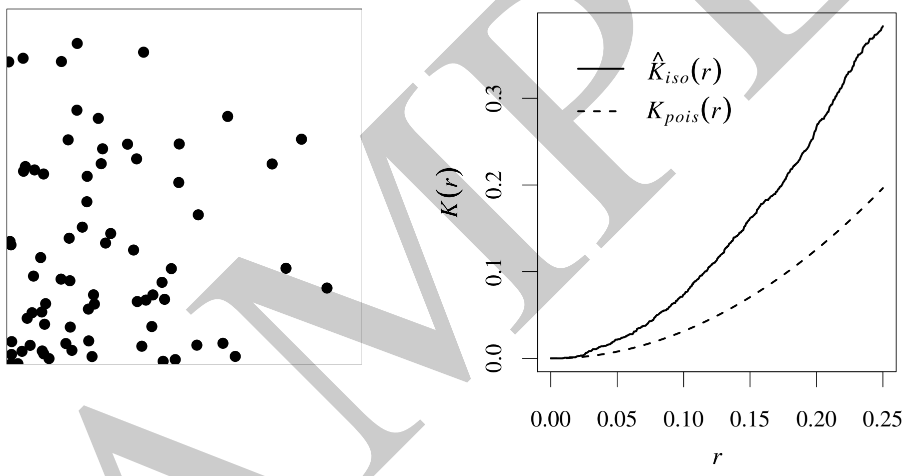

Figure 7.10. *The K-function is fooled by spatial inhomogeneity.* Left: *Inhomogeneous Poisson point pattern.* Right: *Empirical K-function.*

7.3.5.2 Correlation is not causation

There are many possible causes and mechanisms of correlation between points. True spatial regularity could be the result of ecological processes (territorial behaviour, competition for resources), physical forces (repulsion between objects), or human intervention. Apparent spatial regularity can occur as an artefact of the treatment of data, for example, if biological cells of appreciable size are treated as ideal points, or if spatial coordinates are discretised. Spatial clustering does not imply that the points are organised into identifiable 'clusters'; merely that they are closer together than would be expected for a completely random pattern. True spatial clustering could be the result of biological processes (reproduction, contagion), physical forces (electrostatic or magnetic attraction), or space-time history (clustering of meteorite impacts). Apparent clustering can occur if parts of a spatial pattern are obliterated (e.g., native forest partially destroyed by fire) or if objects of one type are physically *repelled* by another type of object. Spatial inhomogeneity is often mistaken for spatial clustering as we mentioned above.

7.3.5.3 Lack of correlation does not prove independence

Correlation is a summary measure of stochastic dependence, but absence of correlation does not necessarily indicate independence. Examples are well known in elementary statistics.

Similarly, there exist point processes whose K-functions are equal to πr^2 and yet the processes are *not* Poisson processes (so that there is dependence amongst the points). Therefore the K-function *does not completely characterise the point process.*

An example is the *cell process* [69]. Space is divided into equal tiles or cells; in each cell we place a random number N of points, according to a probability distribution whose variance is equal to its mean ($\mathbb{E}N = \mathrm{var}N$); the points are positioned independently and uniformly within the cell. The cell process has exactly the same theoretical K-function as the homogeneous Poisson process, but is manifestly different from a Poisson process. The left panel of Figure 7.11 shows a realisation of the cell process, generated by the `spatstat` function `rcell`:

```
  > Xcell <- rcell(nx=15)
```

The right panel shows the empirical K-function, which falsely suggests that the process is Poisson. In fact the cell process would defeat *any* technique based on second moments, because all second-moment quantities for the cell process are identical to those for the Poisson process. Ripley [576, p. 23] commented: "It is a measure of the success of second-order methods that they have been mistakenly assumed to be all-powerful!"

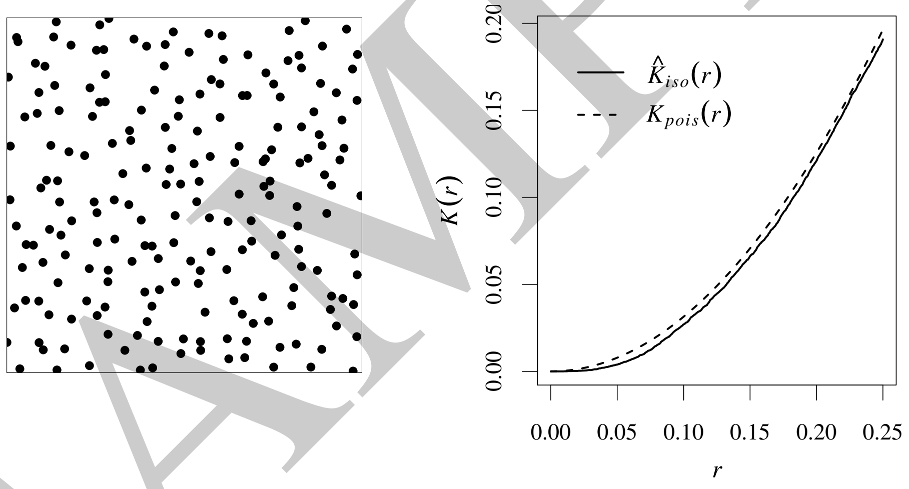

Figure 7.11. *Realisation of Baddeley-Silverman cell process* (Left) *and its empirical K-function* (Right).

7.3.5.4 Spatial scale of interaction

Summary functions like Ripley's K-function convey information across a range of spatial scales. This is an important motivation for using empirical functions, rather than simple numerical summary statistics, and for displaying them graphically.

Many researchers use a graph of the K-function to infer 'the' scale of spatial interaction in a point pattern. This scale is often estimated by reading off the position where the empirical function lies furthest away from the theoretical Poisson value, or furthest outside the region enclosed between the simulation envelopes.

This interpretation is not correct for several reasons. Firstly, 'the scale of interaction' is not a well-defined concept for point processes in general. It is only meaningful for certain point process models, such as Markov point processes (Chapter 13) and some Neyman-Scott cluster processes (Chapter 12).

Secondly, even for point processes which have a well-defined scale of interaction, it is not always true that the greatest deviation in the K-function occurs when r is equal to the scale of interaction. Examples are given in [40, p. 487]. The K-function reflects *correlation* between pairs of points, not direct dependence. Dependence between points at one scale can give rise to correlation between points at another scale.

Thirdly, the K-function is *cumulative*: $K(r)$ accumulates contributions from all distances *less than or equal to* r. If a pattern of emergent seedlings yields $\widehat{K}(r) > \pi r^2$ for the distance $r = 10$ metres, this does not necessarily indicate that seedlings are clustered **at** distances of 10 metres. A plausible explanation is that seedlings are organised in clusters at a much smaller spatial scale, and the cumulative effect is still evident at 10 metres. An alternative, non-cumulative summary statistic is the pair correlation function (Section 7.6). The K-function is optimal for detecting interpoint interaction that occurs equally at all distances up to a certain maximum distance r.

7.4 Edge corrections for the K-function*

This section gives more detail about the edge correction techniques used in estimating the K-function of a point process.

Most users of `spatstat` will not need to know the theory behind edge correction, the details of the techniques, or their relative merits. So long as some kind of edge correction is performed (which happens automatically in `spatstat`), the particular choice of edge correction technique is usually not critical.

However, there is a danger that the *implementations* of edge corrections in some software packages may be incorrect, which could cause substantial bias. There are certainly some misunderstandings about edge corrections that have crept into the applied literature. This section attempts to set the record straight.

For advanced users, understanding the assumptions behind edge correction is helpful in appreciating potential weaknesses of the analysis, and in resolving discrepancies between results. Edge corrections are reviewed in [576, chap. 3], [59, 355].

7.4.1 Estimation without edge effects

We recall that $d_{ij} = \|x_i - x_j\|$ is the distance between a pair of distinct data points x_i, x_j, and

$$t(u, r, \mathbf{x}) = \sum_{j=1}^{n(\mathbf{x})} \mathbf{1}\left\{0 < \|u - x_j\| \le r\right\}$$

is the number of r-neighbours of the location u. An alternative expression for the K-function, that does not involve conditional expectation, is

$$K(r) = \frac{\mathbb{E}\sum_{x \in \mathbf{X} \cap B} t(x, r, \mathbf{X})}{\lambda\, \mathbb{E} n(\mathbf{X} \cap B)} \tag{7.11}$$

holding for a stationary point process $\mathbf{X}$, and for any bounded region B with area $|B| > 0$. Thus $\lambda K(r)$ is the expected number of r-close pairs of points in which the first point falls in B, divided by the expected number of points falling in B.

$*$ Starred sections contain advanced material, and can be skipped by most readers.

A straightforward way to estimate the K-function is suggested by the representation in equation (7.11). Given a point pattern dataset $\mathbf{x}$, we simply replace the numerator and denominator of (7.11) by data-based estimates:

$$\widetilde{K}(r) = \frac{\sum_{x_i \in \mathbf{x} \cap B} t(x_i, r, \mathbf{x})}{\overline{\lambda}\, n(\mathbf{x} \cap B)}. \tag{7.12}$$

The numerator is the total number of r-neighbours of all random points falling in B (an arbitrary set with nonzero area), and in the denominator, $n(\mathbf{x} \cap B)$ is the number of points falling in B, while $\overline{\lambda} = n(\mathbf{x} \cap B)/|B|$ is an estimate of λ.

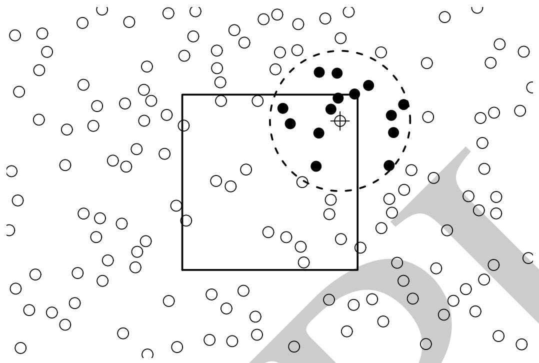

Figure 7.12. *Counting, without edge effects, the number of neighbours of each wildflower within a sampling frame (black rectangle) in a field of wildflowers.*

Figure 7.12 sketches how this estimator could be applied in practice. In a homogeneous field of wildflowers, we have laid out a sampling frame B (black square). Visiting one of the flowers inside the sampling frame, we push a thin stake into the soil near the flower (crosshairs), tied to a string exactly one metre long. Pulling the string taut, we describe a circle around the stake (dashed lines), and count how many other flowers are inside the circle (black dots) *regardless of whether they lie inside or outside the frame*. For the example in Figure 7.12 the count is 14. We have just counted the number of r-neighbours $t(x_i, r, \mathbf{x})$ for one flower x_i, for the distance $r = 1$ metre.

Repeating this process for each flower inside the study area would give us the neighbour counts $t(x_i, r, \mathbf{x})$ for flowers $x_1, \ldots, x_n$, say, where n is the number of flowers in the study area. The average observed neighbour count is $\bar{t}(r) = (1/n)\sum_i t(x_i, r, \mathbf{x})$. Dividing by the estimated intensity $\overline{\lambda} = n/|B|$, where $|B|$ is the area of the study frame B, gives us an estimate of $K(r)$ as in (7.12).

7.4.2 Edge effects

The situation sketched in Figure 7.12 is unusual, because we were able to look outside the sampling frame. In most research studies, this information is lost; points are recorded only if they fall inside the study region; the situation is more like Figure 7.13.

If points are observed only inside a window W, then the estimator (7.12) with $B = W$ is not feasible. The number of points inside a circle of radius r, centred on a point of the process inside W, is *not observable* if the circle extends outside W. This is an *edge effect* problem.

It might be tempting to ignore this problem and use the 'uncorrected' empirical function

$$\widehat{K}_{un}(r) = \frac{|W|}{n(n-1)} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \mathbf{1}\left\{d_{ij} \le r\right\} = |W|\widehat{H}(r), \tag{7.13}$$

the counterpart of (7.3) without the weighting terms $e_{ij}(r)$. However, a simple experiment shows that (7.13) is severely biased as an estimator of $K(r)$. We generate a completely random point

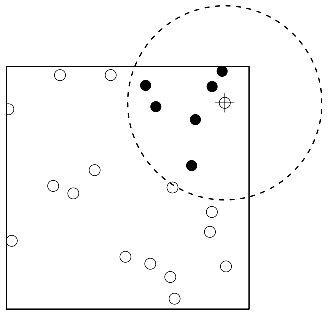

Figure 7.13. *Edge effect problem for estimation of the K-function. If we can only observe the points inside a window W (solid lines), then the number of points inside a circle (dashed lines) of radius r, centred on a point of the process inside W, is not known if the circle extends outside W. The number of points* observed *inside the circle (i.e. counting only points within the window) is typically less than the true number.*

pattern, according to a uniform Poisson process with intensity $\lambda = 100$ in the unit square. The uncorrected function $\widehat{K}_{un}(r)$ is plotted in Figure 7.14 together with the correct K-function $K(r) = \pi r^2$. The uncorrected function is clearly an underestimate of the correct K-function.

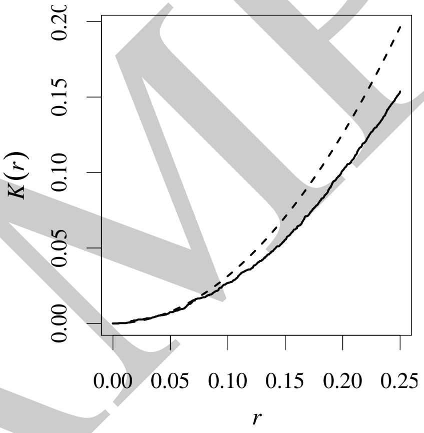

Figure 7.14. *Bias in estimating $K(r)$ due to edge effects. Solid lines: uncorrected estimate $\widehat{K}_{un}(r)$ for a completely random point pattern. Dashed lines: correct K-function $K(r) = \pi r^2$.*

The edge effect bias can be predicted for a given window `W` using the function `distcdf`, which computes the true cumulative distribution function $H(r)$ of the distance between two independent random points in the window.

7.4.3 Border correction

An *edge correction* is a strategy for eliminating the edge effect bias. Edge corrections are surveyed in [576, chap. 3], [59], [355, sec. 4.3.3].

One simple strategy is the *border method*. When estimating $K(r)$ for a particular distance r, we restrict attention to cases where the circle of radius r lies entirely inside the window, so that the edge effect does not occur. See Figure 7.15.

In the numerator and denominator of (7.12) we restrict the summation to data points x_i for which $b(x_i, r)$ lies entirely inside W. For such points, $t(x_i, r, \mathbf{X} \cap W) = t(x_i, r, \mathbf{X})$, so the true number of r-

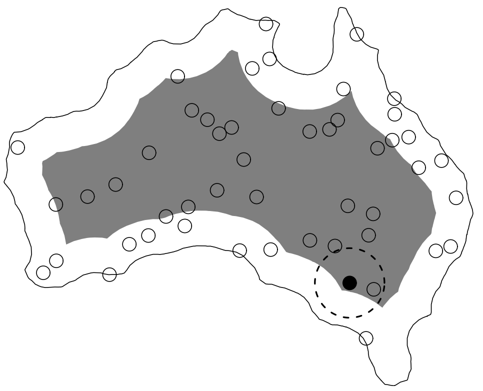

Figure 7.15. *Border method of edge correction for the K-function. When estimating $K(r)$, distances d_{ij} are measured only from points x_i lying at least r units away from the boundary. These are the points falling inside $W_{\ominus r}$ (shaded region).*

neighbours is observable. If $d(u, \partial W)$ denotes the shortest distance from a location u to the window boundary ∂W, then we are restricting attention to data points x_i which satisfy $d(x_i, \partial W) \geq r$. These are the data points which fall in the *eroded set*

$$W_{\ominus r} = W \ominus b(0,r) = \{u \in W : \ d(u, \partial W) \geq r\} \tag{7.14}$$

consisting of locations in W that are at least r units away from the boundary ∂W. The eroded set is shaded in Figure 7.15.

Thus we estimate $K(r)$ by the *border correction estimate*

$$\widehat{K}_{bord}(r) = \frac{\sum_{x_i \in \mathbf{x} \cap W_{\ominus r}} t(x_i, r, \mathbf{x})}{\overline{\lambda}\, n(\mathbf{x} \cap W_{\ominus r})} = \frac{\sum_{i=1}^{n} \mathbf{1}\{b_i \geq r\}\ t(x_i, r, \mathbf{x})}{\overline{\lambda} \sum_{i=1}^{n} \mathbf{1}\{b_i \geq r\}} \tag{7.15}$$

where $b_i = d(x_i, \partial W)$ is the distance from the data point x_i to the window boundary, and $\overline{\lambda}$ is an estimate of the intensity, usually $\overline{\lambda} = n(\mathbf{x} \cap W)/|W|$.

The estimate (7.15) is well defined so long as the denominator is non-zero, that is, so long as $r < \max_i b_i$. The maximum possible value of r for which (7.15) can ever be computed is[3] the *inradius* of W, the radius of the largest disc contained in W, since this is the maximum possible value of $d(\cdot, \partial W)$.

The border correction estimate (7.15) can be justified as a data-based estimate of the right-hand side of (7.11) taking $B = W_{\ominus r}$. This suggests that it will have reasonable statistical properties in sufficiently large datasets. The numerator of (7.15) is an unbiased estimator of the numerator of (7.11), while the denominator of (7.15) is the product of two terms which are unbiased estimators of the corresponding terms in the denominator of (7.11). In large datasets, the border-correction estimate is consistent and approximately unbiased, under reasonable assumptions.

Like many estimators in spatial statistics, the empirical K-function in (7.15) is a slightly biased estimator of the true K-function. It is composed of several terms, each of which is an unbiased estimator of a corresponding theoretical quantity. This implies that the overall bias of the empirical K-function will be small in large samples. Another way to write equation (7.11) is

$$\lambda^2 K(r) = \frac{1}{|B|} \mathbb{E} \sum_{x \in \mathbf{X} \cap B} t(x, r, \mathbf{X}) \tag{7.16}$$

[3]Here and throughout the discussion we assume W is a topologically regular set, meaning that it is the closure of its interior.

so that $\lambda^2 K(r)$ is the expected number of r-close pairs of points with the first point falling in B, divided by the area of B. In essence it is really $\lambda^2 K(r)$ that we need to estimate.

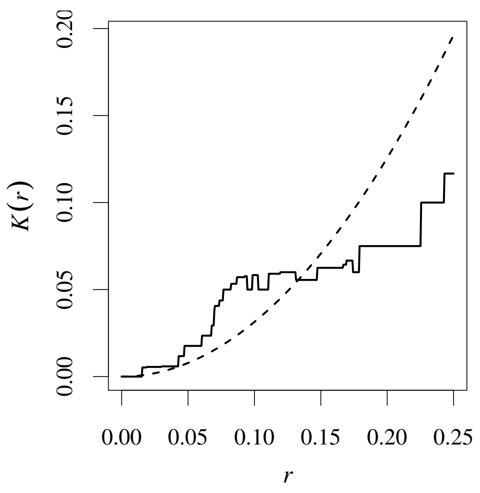

Figure 7.16. *Border correction estimate of the K-function (solid lines) for 20 uniformly random points in the unit square.*

The border correction estimate of the K-function is relatively simple to implement in software for any shape of window, and is fast to calculate. However, in small datasets, it can be inaccurate when compared to other methods, because it discards substantial amounts of data. For example, if W is the unit square and $r = 0.1$, the eroded window $W_{\ominus r}$ is a square of side 0.8 and area 0.64 so that almost 40% of the data is being discarded in order to estimate $K(0.1)$. In small datasets or in unusual situations, the graph of the estimated function $\widehat{K}_{bord}(r)$ can have an erratic trajectory, as shown in Figure 7.16. For large values of r, the denominator of (7.15) is small, magnifying the effect of discrete jumps in the numerator and denominator. Although the true K-function must be an increasing function of r, the border correction estimate need not be so.

When the number of data points is large, the border method is usually preferable to more computationally intensive methods, because the interesting distances r are small, and the loss of statistical efficiency in the border method is negligible.

The border method is a useful general-purpose remedy for edge effects in many spatial problems. It is related to the 'local knowledge principle' of mathematical morphology [622, 65].

7.4.4 Isotropic correction

An alternative approach to edge effects is to regard them as a form of *sampling bias*.

Visualise the entire point process $\mathbf{X}$ extending throughout two-dimensional space. Consider a pair of points x, x' from $\mathbf{X}$, and assume that x falls in the observation window. Given the location of the first point x and the distance $d = \|x - x'\|$, the second point x' must lie somewhere on the circle $b(x,d)$ of radius d centred at x. In general, only part of this circle lies inside the window: see Figure 7.17.

If the point process is also *isotropic* (statistically invariant under rotation, Section 5.6.3), then roughly speaking, the second point x' is equally likely to lie anywhere on this circle. The probability that x' falls inside W is the fraction of length of the circle lying within W,

$$p(x,d) = \frac{\ell(W \cap \partial b(x,d))}{2\pi d}$$

where ℓ denotes length. As d increases, the probability $p(x,d)$ typically decreases. This gives rise to a sampling bias: larger distances are less likely to be observed.

A standard strategy for correcting sampling bias is the *Horvitz-Thompson* estimator [341, 58, 173] introduced on page 181. This strategy can be applied to estimators of the K-function. If each pair of points x_i, x_j is weighted by the reciprocal of the probability $p(x_i, d_{ij})$, we obtain Ripley's *isotropic correction* estimator [572, 569]

$$\widehat{K}_{iso}(r) = \frac{1}{\lambda n} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \mathbf{1}\{d_{ij} \le r\} \frac{1}{p(x_i, d_{ij})}. \tag{7.17}$$

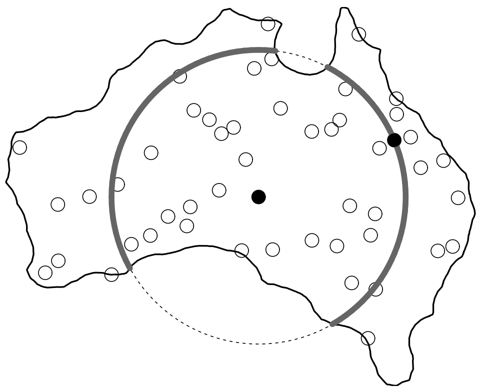

Figure 7.17. *Calculation of sampling bias for the isotropic correction. The contribution from a pair of data points x_i, x_j is determined by drawing a circle through x_j centred at x_i. The fraction p of the circle's* perimeter *which falls inside the observation window (grey line) is measured. The edge correction weight for this pair of points is $1/p$.*

> Warning: Some authors state, incorrectly, that Ripley's isotropic correction uses the fraction of *area* rather than the fraction of *length* of the circle. This error may have found its way into computer code.

The isotropic-correction estimate is a non-decreasing function of r, unlike the border-correction estimate.

The Horvitz-Thompson principle is valid provided each item in the population has a non-zero probability of being sampled. In the isotropic correction, it is possible to have $p(x,d) = 0$ when d is large enough. Let $\widehat{R}_{iso}$ be the smallest distance for which there is some location u in W (not necessarily a data point) where $p(u,d) = 0$. Then $\overline{\lambda}^2 \widehat{K}_{iso}(r)$ is an unbiased estimator of $\lambda^2 K(r)$ for all $r < \widehat{R}_{iso}$. The distance $\widehat{R}_{iso}$ is the circumradius of W, the radius of the smallest circle containing W (or the minimum of the circumradii of the connected components of W, if W consists of several pieces). If W is a rectangle, $\widehat{R}_{iso}$ is equal to half the diagonal of W.

The isotropic estimator can be modified to work for longer distances. For a given distance r, let $W^*(r)$ be the subset of W consisting of locations u where $p(u,r) > 0$. Then the modified estimator of Ohser [518] is

$$\widehat{K}_{iso*}(r) = \frac{1}{\overline{\lambda} n} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \mathbf{1}\{d_{ij} \leq r\} \frac{2\pi d_{ij}}{\ell(W \cap \partial b(x_i, d_{ij}))} \frac{|W|}{|W^*(d_{ij})|} \tag{7.18}$$

and $\overline{\lambda}^2 \widehat{K}_{iso*}(r)$ is an unbiased estimator of $\lambda^2 K(r)$ for all $r < \widehat{R}_{iso*}$. Here $\widehat{R}_{iso*}$ is the smallest distance R such that $|W^*(R)| = 0$, which is equal to the diameter of W if W is a connected set. The extra factor in (7.18) is the reciprocal of the fraction $|W^*(d_{ij})|/|W|$ of window area occupied by $W^*(d_{ij})$. Of course (7.18) agrees with (7.17) for $r < \widehat{R}_{iso}$ so this calculation is only required when $\widehat{R}_{iso} \leq r < \widehat{R}_{iso*}$.

For any $r < \widehat{R}_{iso}$, the double sum in (7.17), namely $\overline{\lambda} n \widehat{K}_{iso}(r)$, is an unbiased estimator of $\lambda^2 |W| K(r)$.

The Ohser modified isotropic correction provides estimates of $K(r)$ for much larger distances r than are possible with the border correction. However, at large distances, the variance of the estimator increases rapidly. This is a well-known problem with the Horvitz-Thompson estimator: if the probability of sampling an item is very small, it will be given a very large weight if it is sampled, so its contribution has large variance.

A pragmatic solution is to restrict the range of r values, and to truncate the edge correction weight in (7.18) so that it never exceeds a specified upper limit. This is the procedure used in `spatstat`.

Stein [625, 626] proposed a more satisfactory solution, in which contributions to the estimator are downweighted if they have large variance. Future versions of `spatstat` will include this estimator.

7.4.5 Translation correction

Another approach to edge correction can be followed if we are not willing to assume the point process is isotropic, or if the computations required for the isotropic correction are too intensive.

Visualise a stationary point process $\mathbf{X}$ (a 'homogeneous field of wildflowers'). Stationarity means that if we apply any translation (shift) vector s to the entire point process, the shifted process $\mathbf{X}+s$ is statistically equivalent to $\mathbf{X}$. Focus on a particular pair of random points x and x' in $\mathbf{X}$. If $\mathbf{X}$ is shifted to $\mathbf{X}+s$, the points x, x' are shifted to new positions $x+s$ and $x'+s$. They are in the same *relative* position: that is, the vector difference $v = x' - x$ from x to x' remains unchanged.

We may imagine that x and x' are joined by an arrow. When $\mathbf{X}$ is shifted to $\mathbf{X}+s$, the arrow's length and direction (given by $v = x' - x$) remain fixed, but the arrow's position is shifted by the vector s.

An arrow will be observed inside the window W only when both its endpoints x, x' fall in W. Intuitively it is clear that a shorter arrow is more likely to fall inside W than a longer arrow, and this gives rise to a sampling bias: the observed arrows are a biased sample of all arrows, with the bias favoring shorter arrows.

Consider the possible positions of the starting point x for which *both* endpoints x and $x' = x+v$ fall inside W. Notice that $x+v \in W$ if and only if x belongs to $W - v$, a copy of the window W shifted by the vector $-v$. Both endpoints of the arrow fall inside W if and only if $x \in W \cap (W - v)$. The region $W \cap (W - v)$ is shaded in Figure 7.18. It is the intersection of the window W with a shifted copy of itself. The probability that the starting point x falls in the shaded region $W \cap (W - v)$ is related to the *area* of this region.

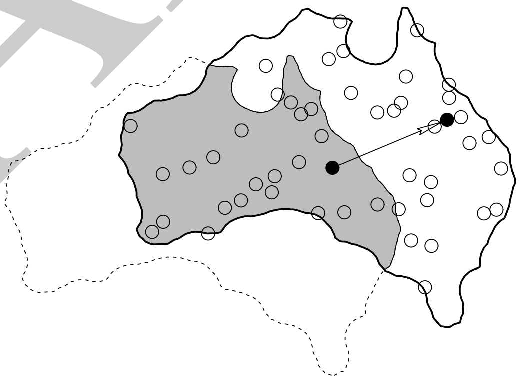

Figure 7.18. *Sampling bias calculation for the translation edge correction. An arrow joining a pair of random points will be observed only if both endpoints fall in the window W. The shaded region shows the possible locations for the start of the arrow which guarantee that both ends of the arrow will be observed.*

Applying the Horvitz-Thompson principle leads to the *translation correction* [518, 520] estimator of the K-function,

$$\widehat{K}_{trans}(r) = \frac{1}{\overline{\lambda}n}\sum_{i=1}^{n}\sum_{\substack{j=1\\j\neq i}}^{n}\mathbf{1}\left\{d_{ij} \le r\right\}\frac{|W|}{|W \cap (W-(x_j-x_i))|}. \tag{7.19}$$

In this estimator, each pair of data points (x_i, x_j) is weighted by the reciprocal of the fraction of window area in which the first data point x_i could be placed so that both points x_i, x_j would be observable (assuming their relative positions were held fixed). This effectively compensates for the sampling bias in observing such pairs.

The edge correction weight in (7.19) can be calculated using simple geometry if W is a rectangle. Otherwise, the most efficient algorithm computes the *set covariance function* $C_W(v) = |W \cap (W-v)|$ for all vectors v on a fine grid using the Fast Fourier Transform (because C is the convolution of two indicator functions) and then extracts the values $C_W(x_j - x_i)$ for each pair of points.

The Horvitz-Thompson principle gives an unbiased estimator provided that each item in the population has a non-zero probability of being sampled. Consequently $\overline{\lambda}^2\widehat{K}_{trans}(r)$ is an unbiased estimator of $\lambda^2 K(r)$ for all $r \le R$, where R is the length of the shortest vector v such that $C_W(v) = 0$. If W is a rectangle, then R is the length of the shortest side.

The translation correction provides estimates of $K(r)$ for larger distances r than are possible with the isotropic correction, but again suffers from inflated variance at these large distances. Again the pragmatic solution used in `spatstat` is to restrict the range of r values, and to truncate the edge correction weight in (7.19) so that it never exceeds a specified upper limit.

If the point process is assumed to be isotropic as well as stationary (Section 5.6.3), the translation correction can be modified to give the *rigid motion correction* [468, 402, 520]

$$\widehat{K}_{rigid}(r) = \frac{1}{\overline{\lambda}n}\sum_{i=1}^{n}\sum_{\substack{j=1\\j\neq i}}^{n}\mathbf{1}\left\{d_{ij} \le r\right\}\frac{|W|}{\overline{C}_W(d_{ij})} \tag{7.20}$$

where $\overline{C}_W$ is the rotational average of C_W, that is, $\overline{C}_W(r)$ is the average value of $C_W(v)$ over all vectors v of length $\|v\| = r$.

7.4.6 Discussion

Edge corrections are discussed and compared in [576, chap. 3], [355, sec. 4.3.3], [59, 640]. The choice of edge correction depends partly on the size of dataset. In small datasets (say, fewer than 100 points) statistical performance is very important, and the methods of choice are the Horvitz-Thompson style weighted edge corrections (translation, isotropic or rigid motion correction); the border method is too imprecise. In moderately large datasets (say 1000 to 10,000 points) the border method performs satisfactorily. In huge datasets no edge correction is necessary, and it is computationally efficient to avoid edge correction. The `spatstat` algorithm for `correction="none"` will handle datasets containing many millions of points.

The choice of edge correction does not, in most instances, seem to be very important, as long as *some* edge correction is applied. (Although it must be applied correctly!) Discrepancies between the results of different edge corrections often tend to indicate unusual features in the data, and usually suggest that the assumption of stationarity is violated.

The accuracy of estimators of the K-function is also affected by the choice of estimator for the squared intensity λ^2. Substituting the natural estimator of λ does not necessarily lead to the most efficient estimator of $K(r)$. Because $\widehat{K}(r)$ is the ratio of an estimator of $\lambda^2 K(r)$ divided by an estimator of λ^2, positive correlation between the numerator and denominator will generally reduce the variability of $\widehat{K}(r)$. Various options for the denominator are canvassed in [355, sec. 4.3.3]. One example favoured in astronomy [317, 401] is

$$\widehat{\lambda}_S(r) = \frac{1}{\overline{C}_W(r)} \sum_i p(x_i, r) \tag{7.21}$$

where, as above, $p(u,r)$ is the length fraction of the circle of radius r centred at location u that lies inside the window W, and $C_w(r)$ is the rotational average of the set covariance function.

7.5 Function objects in `spatstat`

7.5.1 Objects of class `"fv"`

An object of class `"fv"` ('function value table') is a convenient way of storing and plotting several different estimates of the same function.

The value returned by `Kest` is an object of class `"fv"`:

```
> KC <- Kest(cells)
> KC
Function value object (class 'fv')
for the function r -> K(r)
..............................................
       Description
r      distance argument r
theo   theoretical Poisson K(r)
border border-corrected estimate of K(r)
trans  translation-corrected estimate of K(r)
iso    isotropic-corrected estimate of K(r)
..............................................
Default plot formula:  .~r
where "." stands for 'iso', 'trans', 'border', 'theo'
Recommended range of argument r: [0, 0.25]
Available range of argument r: [0, 0.25]
```

An `"fv"` object is a data frame (that is, it also belongs to the class `"data.frame"`) with attributes giving extra information such as the recommended way of plotting the function. One column of the data frame contains evenly spaced values of the distance argument r, while the other columns contain estimates of the value of the function, or the theoretical value of the function under CSR, corresponding to these distance values. The printout for `KC` above indicates that the columns in the data frame are named `r`, `theo`, `border`, `trans`, and `iso`, and explains their contents. For example, the column `iso` contains estimates of the K-function using the isotropic edge correction. The function argument in an `"fv"` object is usually, but not always, called `r`.

7.5.2 Plotting `"fv"` objects

If `f` is an object of class `"fv"`, then `plot(f)` is dispatched to the method `plot.fv`. The default behaviour of `plot(f)` is to generate a plot containing several curves, each representing a different edge-corrected estimate of the same target function, plotted against the distance argument r. For example, the command `plot(Kest(swedishpines))` generates the left panel of Figure 7.19, which shows three different estimates of the K-function for the `swedishpines` dataset, together with the 'theoretical' (expected) curve $K_{pois}(r) = \pi r^2$ for CSR, all plotted against the distance argument r. The legend indicates the meaning of each curve. The main title identifies the function object that was plotted.

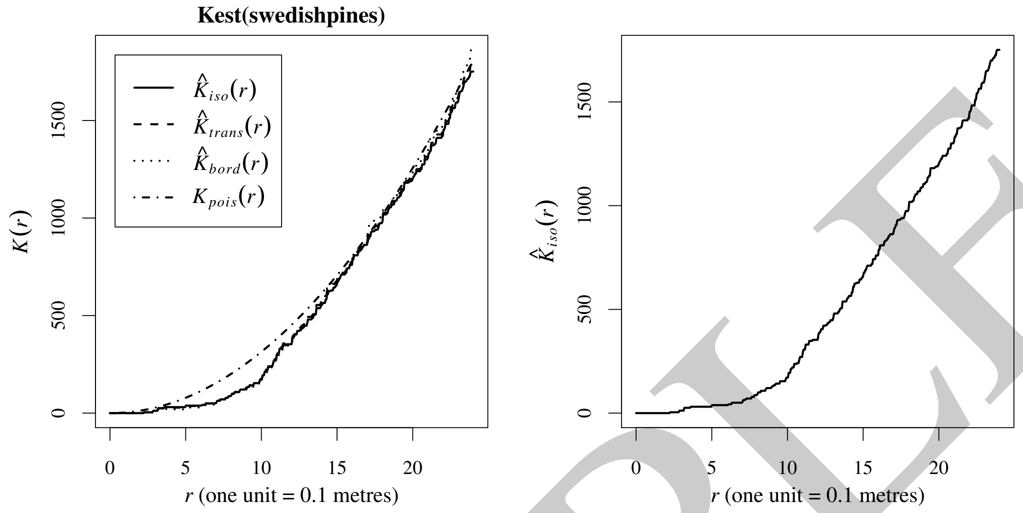

Figure 7.19. *Plots produced by* `plot.fv`*.* Left: *default plot in response to the command* `plot(Kest(swedishpines))`. Right: *plot of isotropic correction estimate $\widehat{K}_{iso}(r)$ against r.*

The return value from `plot.fv` is a data frame containing more detailed information about the meaning of the curves. For the left panel of Figure 7.19, the return value is

```
        lty col                         label                                meaning
iso       1   1   italic(hat(K)[iso](r))   isotropic-corrected estimate of K(r)
trans     2   1 italic(hat(K)[trans](r)) translation-corrected estimate of K(r)
border    3   1  italic(hat(K)[bord](r))       border-corrected estimate of K(r)
theo      4   1       italic(K[pois](r))              theoretical Poisson K(r)
```

Here `lty` and `col` are the graphics parameters controlling the line type and line colour, and `label` is the mathematical notation for each edge-corrected estimate, in the syntax recognised by R graphics functions.

The default plot in the left panel of Figure 7.19 can easily be modified. Set `legend=FALSE` to suppress the legend, and `main=""` to suppress the main title. The legend position is automatically computed to avoid overlap with the plotted curves, but this can be overridden by `legendpos`. For further options see `help(plot.fv)`.

The printout of an `"fv"` object indicates the range of values of `r` in the table as the '`available range`'. It also gives a '`recommended range`' which is generally shorter than the available range. *The default plot of the object will only show the function values over the recommended range* and not over the full range of values available. This is done so that the interesting detail is clearly visible in the default plot. Values outside the recommended range may be unreliable due to increased variance or bias, depending on the edge correction. To change the range of `r` values, use the argument `xlim` in the plot command.

The variables plotted on the x and y axes are determined by the second argument to `plot.fv`, which should be a `formula`, involving the names of columns of the object. The left side of the formula represents what variables will be plotted on the y-axis, and the right side determines the x variable for the plot. For example, in the object `Kest(swedishpines)`, the column named `iso` contains the values of the isotropic correction estimate. The command

```
> Ks <- Kest(swedishpines)
> plot(Ks,  iso ~ r)
```

plots the isotropic correction estimate against r as shown in the right panel of Figure 7.19.

The many uses of formulae for controlling plots were introduced in Section 2.1.10. In `plot.fv`, both sides of the plot formula are interpreted as *mathematical expressions*, so that operators like '+', '-', '*', '/' have their usual meaning in arithmetic. The right-hand side of the formula can be any expression that, when evaluated, yields a numeric vector, and the left-hand side is any expression that evaluates to a vector or matrix of compatible dimensions.

If evaluation of the left-hand side yields a matrix, then each column of that matrix is plotted against the specified x variable as a separate curve. In particular the left-hand side of the formula may invoke the function `cbind` to indicate that several different curves should be plotted. For example, to plot only the translation and isotropic correction estimators and the theoretical curve, an appropriate syntax is: `plot(Ks, cbind(iso, trans, theo) ~ r)` (results not shown).

The plot formula may also involve the names of constants like `pi`, standard functions like `sqrt`, and some special abbreviations listed in Table 7.1. The symbol `.x` represents the function argument, usually named `r`. One of the columns of function values will be designated as the 'best' estimate, for use by some other commands in `spatstat`. This column is identified by the symbol `.y`. Some or all of the columns of function values are designated as 'acceptable' estimates for the default plot, and these are identified by the symbol '`.`'. The default plotting formula is `. ~ .x` indicating that the acceptable estimates will be plotted against the function argument.

`.x`	argument of function
`.y`	best estimate of function
`.`	all estimates of function for default plot
`.a`	all columns of function values
`.s`	upper and lower limits of shading

Table 7.1. *Recognised abbreviations for columns of an* `"fv"` *object. To expand these abbreviations, use* `fvnames(f, ".x")` *and so on.*

A plot formula can be used to specify a transformation that should be applied to the function values before they are displayed. For example, to divide each of the function estimates by the theoretical Poisson value, one could use `plot(Ks, . / theo  ~ r)`. The resulting plot is shown in the left panel of Figure 7.20. Alternatively one could plot the function estimates *against* the Poisson value, using `plot(Ks, . ~ theo)`. This is shown in the right panel of Figure 7.20.

The mathematical labels for the plot axes, and for the individual curves, are constructed automatically by `spatstat` from the plot formula. If the plot formula involves the names of external variables, these will be rendered in Greek where possible. For example, to plot the average number of trees surrounding a typical tree in the Swedish Pines data,

```
> lambda <- intensity(swedishpines)
> plot(Ks, lambda * . ~ r)
```

Here we use the name `lambda` so that it will be rendered as the Greek letter λ in the graphics: the y-axis will be labelled $\lambda K(r)$.

In the discussion of Ripley's paper, Cox [178] proposed that $\widehat{K}(r)$ should be plotted against r^2. This also linearises the plot of the K-function and has some theoretical support. This plot can easily be generated, using the plot formula `. ~ r^2`.

7.5.3 Manipulating `"fv"` objects

An `"fv"` object can be manipulated using the operations listed in Table 7.2.

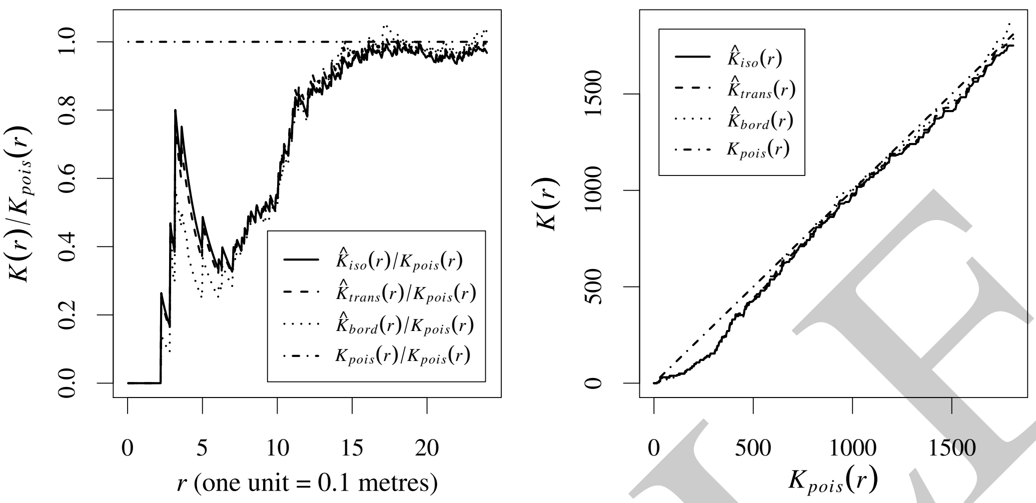

Figure 7.20. Left: *Estimates of $K(r)$ divided by πr^2, plotted against r.* Right: *Estimates of $K(r)$ plotted against πr^2.*

`f`	print a description
`print(f)`	print a description
`plot(f)`	plot the function estimates
`as.data.frame(f)`	strip extra information (returns a data frame)
`f$iso`	extract column named `iso` (returns a numeric vector)
`f[i,j]`	extract subset (returns an `"fv"` object)
`subset(f, ...)`	extract subset (returns an `"fv"` object)
`with(f, expr)`	perform calculations with columns of data frame
`eval.fv(expr)`	perform calculations with several `"fv"` objects
`cbind(f1, f2, ...)`	combine `"fv"` objects `f1, f2, ...`
`bind.fv(f, d)`	combine an `"fv"` object `f` and data frame `d`
`collapse.fv(f1, f2, ...)`	combine several redundant `"fv"` objects
`compatible(f1, f2, ...)`	check whether `"fv"` objects are compatible
`harmonise(f1, f2, ...)`	make `"fv"` objects compatible
`min(f),max(f),range(f)`	range of function values
`Smooth(f)`	apply smoothing to function values
`deriv(f)`	derivative of function
`stieltjes(g,f)`	compute Stieltjes integral with respect to `f`
`as.function(f)`	convert to a function

Table 7.2. *Operations for manipulating an* `"fv"` *object* `f`.

Values of the function

An `"fv"` object is essentially a table, containing the values of the desired function (such as $K(r)$) at a finely spaced grid of values of the function argument r. The function values can be extracted directly from an `"fv"` object since it is also a data frame. A single column of values can be extracted using the `$` operator in the usual way: `Ks$iso` would extract a vector containing the isotropic correction estimates of $K(r)$.

The subset extraction operator '`[`' has a method for `"fv"` objects. This always returns another `"fv"` object, so it will refuse to remove the column containing values of the function argument `r`, for example. To override this refusal, convert the object to a data frame using `as.data.frame` and then use '`[`': the result will be a data frame or a vector.

Commands designed for data frames often work for `"fv"` objects as well. The functions `head` and `tail` extract the top (first few rows) and bottom (last few rows) of a data frame. They also work on `"fv"` objects: the result is a new `"fv"` object containing the function values for a short interval of r values at the beginning or end of the range. The function `subset` selects designated subsets of a data frame using an elegant syntax (see page 101) and this also works on `"fv"` objects. To restrict `Ks` to the range $r \le 0.1$ and remove the border correction,

```
> Ko <- subset(Ks, r < 0.1, select= -border)
```

Convert to a true function

The table of function values can also be converted to a true function in the R language using `as.function`. This makes it easy to evaluate the function at any desired distance r.

```
> Ks <- Kest(swedishpines)
> K <- as.function(Ks)
> K(9)
[1] 129.096
```

By default, the result `K` is a function in R, with a single argument `r` (or whatever the original function argument was called). The new function accepts numeric values or numeric vectors of distance values, and returns the values of the 'best' estimate of the function, interpolated linearly between entries in the table. If one of the other function estimates is required, use the argument `value` to `as.function` to select it. Several estimates can be chosen:

```
> K <- as.function(Ks, value=".")
> K(9)
[1] 129.096
> K(9, "trans")
[1] 130.4998
```

Calculating with columns

To manipulate or combine one or more columns of data in an `"fv"` object, it is typically easiest to use `with.fv`, a method for the generic `with`. For example:

```
> Kr <- Kest(redwood)
> y <- with(Kr, iso - theo)
> x <- with(Kr, r)
```

The results `x` and `y` are numeric vectors, where `x` contains the values of the distance argument r, and `y` contains the difference between the columns `iso` (isotropic correction estimate) and `theo` (theoretical value for CSR) for the K-function estimate of the redwood seedlings data. For this to work, we have to know that `Kr` contains columns named `r`, `iso` and `theo`. Printing the object will reveal this information, as would typing `names(Kr)` or `colnames(Kr)`.

The general syntax is `with(X, expr)` where `X` is an `"fv"` object and `expr` can be any expression involving the names of columns of `X`. The expression can include functions, so long as they are capable of operating on numeric vectors. The expression can also involve the abbreviations listed in Table 7.1. Thus: `Kcen <- with(Kr, . - pi*r^2)` subtracts the 'theoretical' value from all the available edge correction estimates. In this case the result `Kcen` is an `"fv"` object. You can also get a result which is a vector or single number:

```
> with(Kr, max(abs(iso-theo)))
[1] 0.04945199
```

Calculating with several "fv" objects

To manipulate or combine several "fv" objects, spatstat provides the special function eval.fv. Its argument should be an expression involving *the names of* the "fv" objects which are to be combined. For example, to find the difference between the K-functions of the redwood and cells datasets,

```
> K1 <- Kest(redwood) ; K2 <- Kest(cells)
> DK <- eval.fv(K1-K2)
```

The result DK is another "fv" object. Note that something like eval.fv(Kest(redwood) - Kest(cells)) would not work, because redwood and cells are not "fv" objects.

The objects in the expression should be 'compatible' in the sense that they have the same column names, and the same vector of r values. However, eval.fv will attempt to reconcile incompatible objects. (The spatstat generic function compatible determines whether two or more objects are compatible, and the generic function harmonise makes them compatible, if possible.)

Objects of class "fv" are data frames of function values, with extra attributes which explain the function values and determine how they are plotted. The function eval.fv performs arithmetic on the columns of function values, but also manipulates these extra attributes, so that when the result of eval.fv is plotted, the labels on the plot are constructed in a meaningful way.

7.6 The pair correlation function

7.6.1 The pair correlation function of a point process

A plot of the K-function can be difficult to interpret correctly in terms of the behaviour of the point process, because of its 'cumulative' nature (Section 7.3.5). The value of $K(r)$ contains contributions for all interpoint distances *less than or equal to* r.

An alternative tool is the **pair correlation function** $g(r)$ which contains contributions only from interpoint distances *equal to* r. In two dimensions, it can be defined by

$$g(r) = \frac{K'(r)}{2\pi r} \tag{7.22}$$

where $K'(r)$ is the derivative of the K-function with respect to r. See [355, pp. 218–223, 232–244]. The pair correlation function has a long history in astronomy [204, 317, 401, 552], mostly for three-dimensional patterns. It has also arisen independently in geographical analysis and other fields [285].

In geometric terms, $K(r)$ is defined by drawing a circle of radius r centred on a point of the point process, and counting the number of other points falling in the circle (see left panel of Figure 7.21). To define $g(r)$, draw two concentric circles of radius r and $r+h$, where h is a small increment of distance, and count only those points falling in the ring between the two circles (see right panel of Figure 7.21). This captures the interpoint distances d_{ij} that lie in the narrow range between r and $r+h$. The expected count of such distances is $\lambda K(r+h) - \lambda K(r)$. We standardise the expected count by dividing it by the expected value for complete spatial randomness, which is $\lambda\pi(r+h)^2 - \lambda\pi r^2$, yielding

$$g_h(r) = \frac{\lambda K(r+h) - \lambda K(r)}{\lambda\pi(r+h)^2 - \lambda\pi r^2} = \frac{K(r+h) - K(r)}{2\pi rh + \pi h^2}.$$

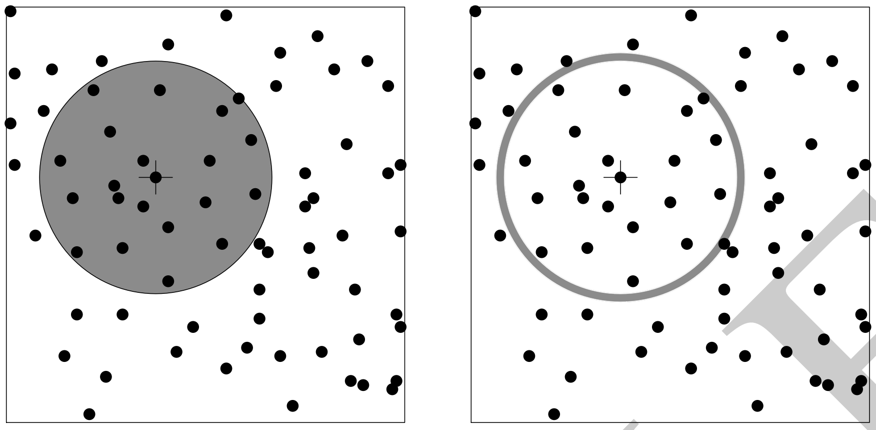

Figure 7.21. *Geometry of the K-function (*Left*) and the pair correlation function (*Right*).*

When h is very small, πh^2 becomes negligible, and $(K(r+h)-K(r))/h$ becomes the derivative of K, so that $g_h(r)$ becomes the pair correlation function (7.22).

The concept of the pair correlation function has also been reinvented in different guises. The *O-ring statistic* [700] is essentially $K(r+h)-K(r)$, where now the ring thickness h is an algorithm parameter that is held fixed and should not be too small. The O-ring statistic is approximately $hg(r)$. The *K density* [247] is defined as the derivative of the K-function, which is therefore exactly equal to a multiple of the pair correlation function, $K'(r) = 2\pi r\, g(r)$. Many writers seem to be unaware of the existence of the pair correlation function and its long history in astronomy and other fields.

Imagine the observation window is divided into a fine grid of pixels. The probability that a given pixel contains a random point is $p \doteq \lambda a$ where a is the pixel area. Here we use the symbol $\doteq$ to mean that the quantities are equal when they are very small.

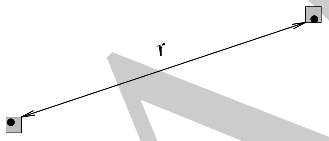

Figure 7.22. *Interpretation of the pair correlation $g(r)$ as the (normalised) probability that two pixels, a distance r apart, both contain random points.*

Now consider two pixels with centres u and v, separated by a distance r. See Figure 7.22. Let $p_2(u,v)$ be the probability that *both* pixels contain random points. If the process is Poisson, then events in these pixels are independent, and $p_2(u,v) = p^2$. For any stationary and isotropic point process, it turns out that

$$g(r) \doteq \frac{p_2(u,v)}{p^2}. \tag{7.23}$$

That is, *$g(r)$ is the probability of observing a pair of points of the process separated by a distance r, divided by the corresponding probability for a Poisson process.* In other words, if U and V are two 'infinitesimal' regions with areas du and dv, respectively, separated by a distance r then

$$\mathbb{P}\{\mathbf{X} \text{ has a point in } U \text{ and a point in } V\} \doteq \lambda^2 g(r)\, \mathrm{d}u\, \mathrm{d}v \tag{7.24}$$

where λ is the intensity of the process.

The value $g(r) = 1$ is consistent with complete spatial randomness, because of the way the function has been standardised. A value $g(r) < 1$ indicates that interpoint distances equal to r are less frequent than would be expected for a completely random process, so this suggests regularity. A value $g(r) > 1$ indicates that this interpoint distance is more frequent than expected for a completely random pattern, which suggests clustering. These interpretations are less ambiguous than the corresponding statements for the K-function, because they refer only to interpoint distances equal to r.

Notice that the value $g(r)$ is not a correlation in the sense used by statisticians. The correlation between two random variables is a number standardised to lie in the range between -1 and $+1$, with the value 0 indicating a lack of correlation. It turns out to be impractical to standardise the correlation structure of point processes in this way. Instead, $g(r)$ is the kind of correlation often used in physics: the possible values range from 0 to infinity, and the value 1 is associated with a lack of correlation.

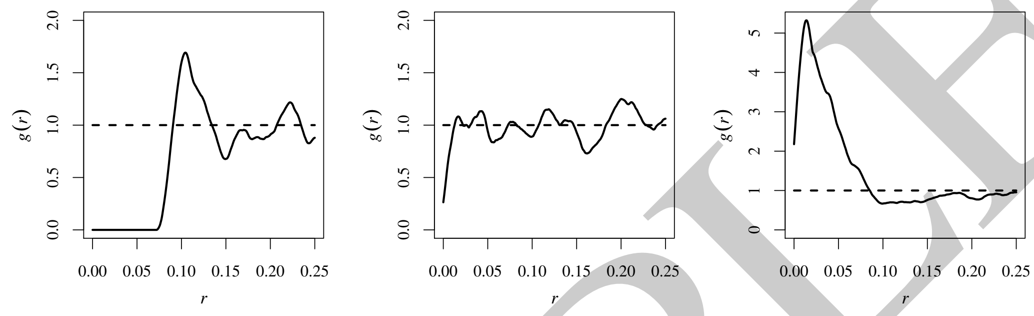

Figure 7.23. *Empirical pair correlation function for each of the three patterns in Figure 7.1.*

Figure 7.23 shows estimates of the pair correlation function for the three archetypal point patterns. Compare these with the empirical K-functions shown in Figure 7.7 on page 206.

The left panel of Figure 7.7 shows that the K-function for the regular pattern is below the Poisson curve for all distances r, suggesting (if we are not careful) that there is inhibition at *all* distances. The plot of the pair correlation has a different message: $g(r)$ is below the Poisson value, $g(r) < 1$, for distances r up to 0.07 units, then it climbs steeply to a value *greater than* the Poisson value, $g(r) > 1$ at about $r = 0.10$ units, before declining back to 1. In fact $g(r)$ is zero for $r \leq 0.07$ because this extremely regular pattern has no interpoint distances shorter than 0.07 units. The reason for the apparent discrepancy between g and K is clear: short interpoint distances are completely absent in this pattern, and this deficit affects the cumulative count of pairwise distances in the K-function, even at much larger distances r.

The K-function and the pair correlation function g are mathematically interrelated: g can be obtained from K through (7.22), and K can be obtained from g by the reverse relationship

$$K(r) = 2\pi \int_0^r s g(s) \, \mathrm{d}s. \tag{7.25}$$

One may ask why anyone would use the K-function, if the pair correlation function g is easier to interpret. The main reason is that K seems to perform better as a basis for statistical inference (such as hypothesis tests).

Also, in theory g does not necessarily exist: it is essentially the derivative of the K-function, and this does not exist if the K-function has jumps. For example, for a regular grid of points (randomly translated to make it a stationary point process), with points spaced at multiples of s units, the K-function has a jump at $r = s$ and at $r = 2s$, $r = s\sqrt{2}$, and so on. This process has no pair correlation function. This example could be realistic for some crystalline structures.

The practical interpretation of the pair correlation function is discussed in [355, sec. 4.3.4].

7.6.2 Estimating the pair correlation function

A popular way to estimate the pair correlation function is by kernel smoothing [635, p. 126], [242, 257, 629, 637, 453, 531]. In very large samples, histogram-based methods could be used, and these are common in astronomical applications.

Taking any edge-correction estimator of the K-function of the general form (7.3), suppose we replace the indicator $\mathbf{1}\left\{d_{ij} \le r\right\}$ by a kernel term $\kappa(d_{ij} - r)$ to obtain a smooth estimate of $K'(r)$, then plug into the formula (7.22) for the pair correlation function, to obtain the 'fixed-bandwidth' kernel estimator [638, eq. (15.15), p. 284]

$$\widehat{g}(r) = \frac{|W|}{2\pi r n(n-1)} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \ne i}}^{n} \kappa_h(r - d_{ij}) e_{ij}(r) \tag{7.26}$$

where $d_{ij} = \|x_i - x_j\|$ is the interpoint distance between the ith and jth data points, $e_{ij}(r)$ is an edge correction weight, and κ_h is the smoothing kernel, with smoothing bandwidth $h > 0$.

The kernel κ_h is a rescaled version of a template kernel κ,

$$\kappa_h(x) = \frac{1}{h}\kappa\left(\frac{x}{h}\right) \tag{7.27}$$

where κ is any chosen function that is a probability density over the real line with mean 0. For example, κ could be the standard normal (Gaussian) density, so that κ_h would be the normal density with mean 0 and standard deviation h. The convention followed in R and in `spatstat` is that the template κ has standard deviation 1, so that κ_h has standard deviation h. The 'smoothing bandwidth" is the standard deviation of the smoothing kernel. (Note that other writers may use a different convention.)

The usual choice of smoothing kernel for pair correlation functions is the *Epanechnikov kernel* with half-width w,

$$\varepsilon_w(x) = \frac{3}{4w}\left(1 - \frac{x^2}{w^2}\right)_+ \tag{7.28}$$

where $(x)_+ = \max(0, x)$. The kernel is a quadratic function truncated to the interval $[-w, w]$. The standard deviation of $\varepsilon_w(x)$ is $h = w/\sqrt{5}$, so the kernel of bandwidth h is $\kappa_h(x) = \varepsilon_{h\sqrt{5}}(x)$.

To compute (7.26) in practice we must choose a bandwidth value h, and as is common in data analysis, this choice involves a compromise between bias and variability. For excessively large bandwidth h, variability will be well controlled but the shape of the function $\widehat{g}$ will be oversmoothed, so that important detail in the pair correlation function may be missed. For excessively small bandwidth, the estimates $\widehat{g}(r)$ will have high variance, and the shape of $\widehat{g}$ will be erratic.

The standard rule of thumb [355, p. 236] is to take the half-width w of the Epanechnikov kernel (7.28) to be $w = c/\sqrt{\widehat{\lambda}}$ where c is a constant between 0.1 and 0.2. This corresponds to taking the bandwidth

$$h = \frac{c'}{\sqrt{\widehat{\lambda}}} \tag{7.29}$$

where $c' = c/\sqrt{5}$ is between 0.045 and 0.090. Alternative bandwidth selection and bias correction methods are discussed in [300], [355, p. 230 ff.]. The variance of $\widehat{g}(r)$ is discussed in [355, p. 232 ff.], [632]. The choice of denominator (related to estimation of the intensity) is also important to controlling the variance, as mentioned in Section 7.4.6.

To compute the estimated pair correlation function in `spatstat` use `pcf`.

```
> g <- pcf(cells)
> plot(g)
```

The function `pcf` is generic, with a method for point patterns. By default, `pcf.ppp` uses the estimator (7.26), with the Epanechnikov kernel, with bandwidth h selected by Stoyan's rule of thumb (7.29) with $c = 0.15$ so that $c' = 0.067$.

In analysing a point pattern dataset, attention is often focused on the behaviour of $g(r)$ for small r, which reflects the correlation between close pairs of points. Unfortunately, estimation of $g(r)$ at small distances is intrinsically difficult, because geometry dictates that there will typically be few observations at small distances. The estimator (7.26) has poor performance at small distances. Theoretically, as r approaches zero, the variance of $\widehat{g}(r)$ becomes infinite, for many point processes including CSR. The left panel of Figure 7.24 shows a real example where $\widehat{g}(r)$ has an infinite asymptote at $r = 0$. These problems are due to the factor $1/r$ in (7.26). An alternative estimator with usually better performance is

$$\widehat{g}(r) = \frac{1}{2\pi} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{\kappa_h(r - d_{ij})}{d_{ij}} e_{ij}(r) \tag{7.30}$$

in which the contribution to $\widehat{g}(r)$ from an observed interpoint distance d_{ij} involves a factor $1/d_{ij}$ rather than $1/r$. This estimator can be invoked by specifying `divisor="d"` in `pcf.ppp`. The right panel of Figure 7.24 shows this estimator computed for the same real example.

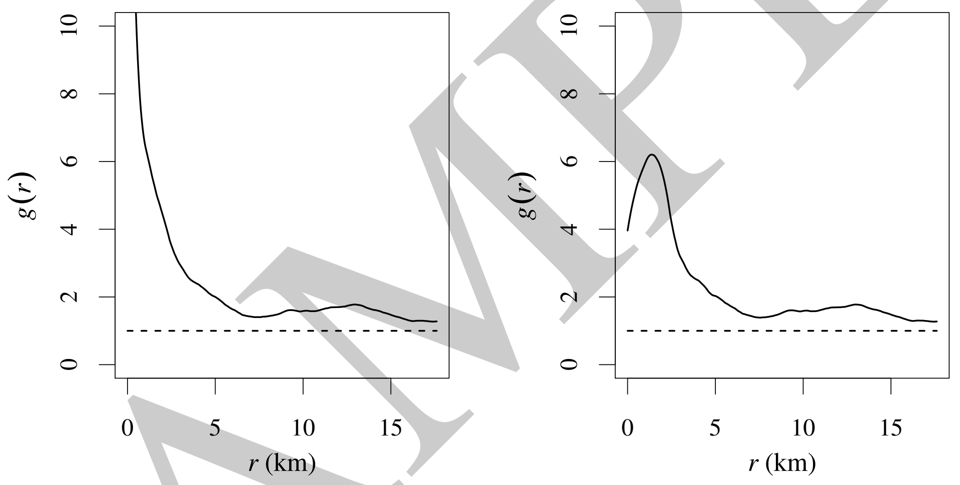

Figure 7.24. *Estimates of the pair correlation function of the Queensland copper deposits of Figure 1.11, using (*Left*) the divide-by-r estimator* (7.26) *and (*Right*) the divide-by-d estimator* (7.30).

Another useful trick applies when an estimate of $K(r)$ is already available: then we can calculate $g(r)$ from $K(r)$ through equation (7.22). This is particularly useful in large datasets, where direct estimation of $g(r)$ can be time-consuming. The conversion of $K(r)$ to $g(r)$ is done by the method `pcf.fv`, which computes the derivative by numerical differentiation using spline smoothing. Effectively the function $K(r)$ is first smoothed by fitting polynomial approximations to $K(r)$ over successive ranges of r values. The derivatives of the polynomials are then computed, using elementary calculus.

```
> K <- Kest(shapley)
> g <- pcf(K, spar=0.5)
```

Since `K` is an object of class `"fv"`, the last command is dispatched to `pcf.fv`. Additional arguments to `pcf.fv`, such as the argument `spar` above, are passed to the function `smooth.spline` in the `stats` package, which performs the spline smoothing. Using a little algebra, we may apply smoothing either to $K(r)$ or to the transformed functions $K(r)/(2\pi r)$, $K(r)/(\pi r^2)$, or $L(r) = \sqrt{K(r)/\pi}$, by specifying `method="a"`, `"b"`, `"c"` or `"d"`, respectively. Methods `"b"` to `"d"` offer improved performance at small distances r. The default is `method="c"` which seems to perform best overall. See `help(pcf.fv)` for more information.

7.7 Standard errors and confidence intervals

Statistical estimates should be accompanied by a measure of accuracy (such as a standard error) or uncertainty (such as a confidence interval). This is not straightforward for the K-function: there is no simple expression for the standard error or variance of $\widehat{K}(r)$, even in a Poisson process, because the contributions to $\widehat{K}(r)$ from different data points x_i are not independent.

7.7.1 Variance under CSR

Approximations to the standard error of $\widehat{K}(r)$ *assuming complete spatial randomness* are available, using the techniques of U-statistics [576, 518, 433, 225]. Ripley [576] gave an approximate variance estimate for the isotropic corrected K-function estimator,

$$\operatorname{var}\widehat{K}_{iso}(r) \approx 2\left(\frac{A}{n-1}\right)^2 \left(\frac{\pi r^2}{A} + \frac{0.96Pr^3}{A^2} + 0.13\frac{nPr^5}{A^3}\right) \tag{7.31}$$

where n is the number of data points and A and P are the area and perimeter length of the window W. Lotwick and Silverman [433] gave a more intricate and accurate approximation for the case of a rectangular window. These approximations are chiefly useful for predicting the accuracy that can be expected from a survey region of given size and shape. They are computed by `Kest` if the argument `var.approx=TRUE` is given. For the L-function, the corresponding approximations for $\operatorname{var}\widehat{L}_{iso}(r)$ are calculated by the delta method.

7.7.2 Block bootstrap

Bootstrap methods [250, 340] make it possible to estimate variance from the data, without assuming the Poisson model. A very simple version of this approach is *block variance estimation*. Assuming the window W is a rectangle, divide it into equal quadrats $B_1, \ldots, B_m$. Suppose our estimator of $K(r)$ is (7.3). For a particular quadrat B, let

$$\widehat{K}(r,B) = \frac{m|W|}{n(n-1)} \sum_{x_i \in B} \sum_{j \neq i} \mathbf{1}\left\{d_{ij} \leq r\right\} e_{ij}(r) \tag{7.32}$$

be an estimate of $K(r)$ based only on the pairs (x_i, x_j) for which x_i falls in the quadrat B, while x_j may lie anywhere in W. The usual estimate $\widehat{K}(r)$ is the average of the estimates $\widehat{K}(r,B_k)$ over all $k = 1, \ldots, m$. We compute the pointwise sample mean, sample variance, and sample standard deviation of these estimates, yielding an elementary bootstrap estimate [225, eq. (4.21), p. 52] of the standard error for $\widehat{K}(r)$. Assuming $\widehat{K}(r)$ is approximately normally distributed, we can then calculate pointwise 95% confidence intervals for the true value of $K(r)$.

The block variance estimate is computed by the `spatstat` function `varblock`:

```
> swp <- rescale(swedishpines)
> Kvb <- varblock(swp, Kest, nx=3, ny=3)
> plot(Kvb)
```

The result is shown in the left panel of Figure 7.25. For a fixed value of r, the grey shading gives a 95% confidence interval for the true value of $K(r)$. Formally this means that, in 95% of outcomes of this procedure, the random interval computed here would embrace the true value of $K(r)$.

Notice that the confidence interval becomes wider as r increases, because the variance of $\widehat{K}(r)$ is roughly proportional to r^2. If we use $L(r)$ instead of $K(r)$, the variance will be stabilised: $\mathrm{var}\,\widehat{L}(r)$ is approximately constant as a function of r. The confidence intervals will then have roughly constant width.

An important caveat is that these are *pointwise* calculations (i.e. performed separately for each value of distance r) yielding *pointwise* confidence intervals (i.e. valid only for a single value of r at a time). Our confidence that the true K-function lies *entirely inside* the grey shaded region, over all values of r, is vastly less than 95%. A solution to this problem is explained below.

7.7.3 Confidence intervals from Loh's bootstrap

A more flexible bootstrap approach was developed by Loh [430]. First we decompose the empirical K-function (7.3) into the contributions from each individual data point x_i,

$$\widehat{K}(r, x_i) = \frac{|W|}{n-1} \sum_{j \neq i} \mathbf{1}\left\{d_{ij} \leq r\right\} e_{ij}(r), \qquad \text{for } i = 1, \ldots, n \tag{7.33}$$

called the **local** K-functions. The usual estimate $\widehat{K}(r)$ is the average of the estimates $\widehat{K}(r, x_i)$ over all $i = 1, \ldots, n$. We could again calculate the pointwise sample mean and sample variance of these local K-functions. Instead, Loh's approach is to resample from the suite of local K-functions. Choose an independent random sample of size n, with replacement, from the numbers 1 to n. If the sample is $i_1, \ldots, i_n$, we calculate the average of the corresponding local K-functions,

$$K^*(r) = \frac{1}{n} \sum_{k=1}^{n} \widehat{K}(r, x_{i_k}). \tag{7.34}$$

This gives a resampled version of $\widehat{K}(r)$. Repeating this procedure a large number N of times, we obtain a bootstrap sample of the distribution of $\widehat{K}(r)$. Pointwise confidence intervals can then be computed directly from the observed quantiles of the bootstrap sample. This calculation is performed by the `spatstat` function `lohboot`:

```
> Kloh <- lohboot(swp, Kest)
> plot(Kloh)
```

The result is shown in the right panel of Figure 7.25. The default behaviour is to compute a pointwise 95% confidence interval.

The grey-shaded intervals in Figure 7.25 are pointwise confidence intervals. Ideally we would prefer a *global* confidence interval or confidence band, having a 95% chance of containing the entire graph of the true K-function. This is possible in Loh's approach. First we replace $K(r)$ by $L(r) = \sqrt{K(r)/\pi}$ to stabilise the variance. For each resampled L-function $L^*(r)$, we calculate the deviation

$$D^* = \max_{0 \leq r \leq r_{max}} |L^*(r) - \widehat{L}(r)|,$$

the maximum vertical separation between the graphs of $L^*(r)$ and $\widehat{L}(r)$ over the interval of r values up to a nominated maximum distance r_{max}. If there are N resampled L-functions, this gives N deviation values; we take the 95th percentile of these deviations, say $D_{0.95}$, and construct the **global confidence band** with boundaries

$$\begin{aligned} L_-(r) &= \widehat{L}(r) - D_{0.95} \\ L_+(r) &= \widehat{L}(r) + D_{0.95}. \end{aligned} \tag{7.35}$$

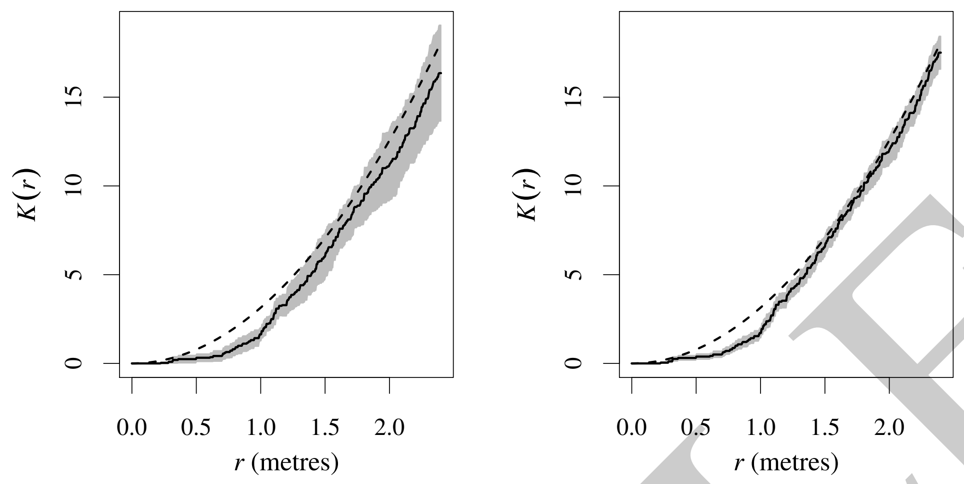

Figure 7.25. *Pointwise 95% confidence intervals for the true K-function of the Swedish Pines using (*Left*) the spatial block bootstrap* `varblock`*, and (*Right*) Loh's bootstrap* `lohboot`*. Solid lines: estimated K-function. Dashed lines: the theoretical K-function under CSR, i.e.,* $K(r) = \pi r^2$.

This has the desired interpretation that, in 95% of outcomes of this procedure, the true L-function will lie entirely between the two limits $L_-(r), L_+(r)$. This procedure is performed by `lohboot` with the argument `global=TRUE`:

```
> Lg <- lohboot(swp, Lest, global=TRUE)
```

A global confidence band for $K(r)$ can then be obtained by back-calculation:

```
> Kg <- eval.fv(pi * Lg^2)
```

The results are plotted in Figure 7.26.

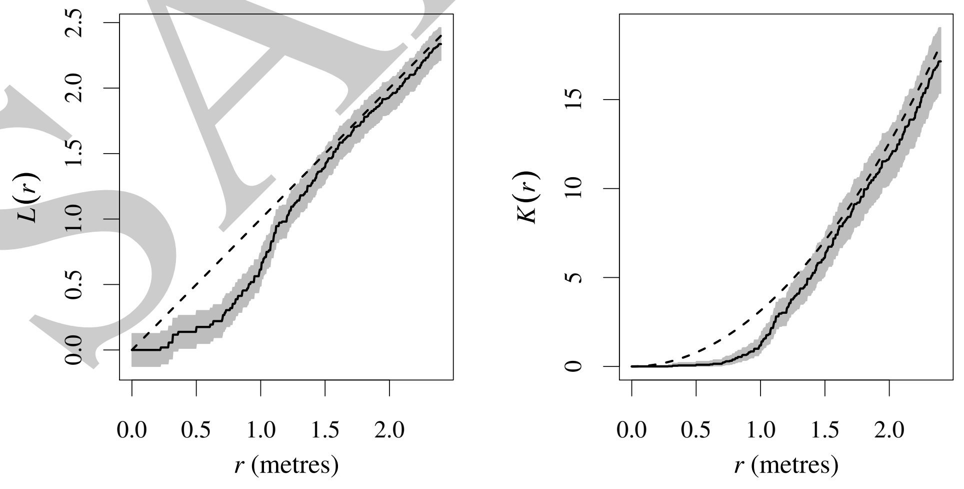

Figure 7.26. *Global 95% confidence interval for the true L-function (*Left*) and back-transformed confidence interval for the K-function (*Right*) of the Swedish Pines, using Loh's bootstrap.*

7.8 Testing whether a pattern is completely random

The K-function estimated from a point pattern dataset, $\widehat{K}(r)$, is often compared graphically with the theoretical K-function for the homogeneous Poisson process, $K_{pois}(r) = \pi r^2$, serving as the benchmark of a completely random pattern. In the examples above, large discrepancies between $\widehat{K}(r)$ and $K_{pois}(r)$ suggested that the Swedish Pines are not completely random. However, even with a completely random pattern, we will never obtain perfect agreement between $\widehat{K}$ and K_{pois}, because of random variability. We need a rule for deciding whether the difference between $\widehat{K}$ and K_{pois} is large enough (relative to the scale of variability) to convince us that the point process is not completely random: that is, whether the discrepancy between $\widehat{K}$ and K_{pois} is *statistically significant*. This rule is a *significance test* or *hypothesis test*. The principles of statistical tests are explained in Chapter 10; here we explain the basic idea and show how to conduct the test in `spatstat`.

7.8.1 Pointwise envelopes

In order to assess the statistical significance of deviations between the observed K-function and the theoretical K-function for CSR, essentially we need to know how much variability should be expected in the estimate $\widehat{K}(r)$ if the pattern is completely random.

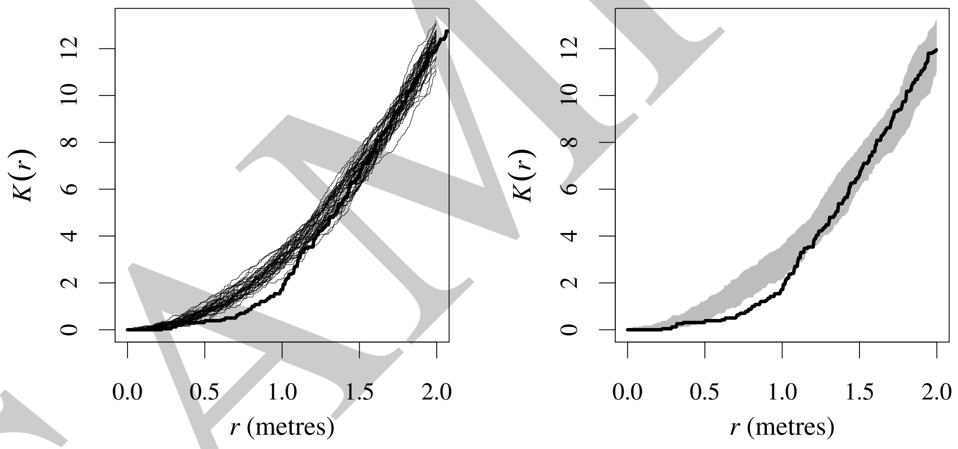

Figure 7.27. *Construction of envelopes of the K-function.* Left: *the K-function estimate of the Swedish Pines dataset (thick line), and of 39 simulated realisations of CSR with the same number of points (thin lines).* Right: *grey shading between the upper and lower* envelopes *(maximum and minimum values) of the K-functions of the simulated patterns.*

The left panel of Figure 7.27 shows the K-function estimated from the rescale `swedishpines` dataset (thick line), and also the K-functions of 39 simulated realisations of CSR with the same number of points (thin lines). Each thin line could have been generated by

```
plot(Kest(runifpoint(npoints(swp), Window(swp))))
```

The spread of the thin lines typifies the variability in $\widehat{K}(r)$ that we should expect if the pattern is completely random with the same number of points as the data. Clearly, the estimate of $K(r)$ for the Swedish Pines data lies outside this range, for some values of r.

In the right panel of Figure 7.27 we have shaded the region between the highest and lowest lines, that is, the minimum and maximum values of the K-functions of the simulated patterns, called the upper and lower *envelopes* of the simulated functions.

The envelope plot can be interpreted as a statistical significance test, as follows. Without peeking at the data, choose a particular distance r, say $r^* = 1$ metre. If the `swedishpines` data were completely random, then the data and simulated patterns would be statistically equivalent, since they would all be completely random patterns with the same number of points. Consequently the values of $\widehat{K}(r^*)$ for the data and simulated patterns would be statistically equivalent numbers. By symmetry, since there are 40 values altogether (one data value and 39 simulated values), there would be a chance of 1 in 40 that the value for the data is the smallest, i.e. that the data value is smaller than all of the simulated values. Similarly there would be 1 chance in 40 that the data value is larger than all the simulated values. This makes 2 chances in 40, or a probability of $1/20 = 0.05$ that the data value lies outside the range of the simulated values purely by chance. This has in fact occurred in Figure 7.27, so we could declare the result to be statistically significant with a p-value of 0.05.

This is an example of a *Monte Carlo test*, a test based on simulations from the null hypothesis, using a symmetry principle. A full explanation of Monte Carlo tests is given in Chapter 10.

The `spatstat` function `envelope` computes simulation envelopes. The right panel of Figure 7.27 was generated by the commands

```
> E <- envelope(swp, Kest, nsim=39, fix.n=TRUE)
> plot(E)
```

The result of `envelope` is an object of class `"envelope"` which also belongs to the class `"fv"`. It can be plotted, printed, and manipulated using the tools for `"fv"` objects described in Section 7.5. See Table 7.2 on page 223. The print method gives a lot of detail:

```
> E
Pointwise critical envelopes for K(r)
and observed value for 'swp'
Edge correction: "iso"
Obtained from 39 simulations of CSR
Alternative: two.sided
Significance level of pointwise Monte Carlo test: 2/40 = 0.05
.......................................................
     Description
r    distance argument r
obs  observed value of K(r) for data pattern
theo theoretical value of K(r) for CSR
lo   lower pointwise envelope of K(r) from simulations
hi   upper pointwise envelope of K(r) from simulations
.......................................................
Default plot formula:  .~r
where "." stands for 'obs', 'theo', 'hi', 'lo'
Columns 'lo' and 'hi' will be plotted as shading (by default)
Recommended range of argument r: [0, 2.4]
Available range of argument r: [0, 2.4]
Unit of length: 1 metre
```

Arguments to `envelope` allow us to specify the data pattern, the summary function to be used, the number of simulations, one-sided or two-sided tests, and so on. Instead of testing CSR, we can use `envelope` to test essentially any null hypothesis. Full details of the `envelope` function are given in Section 10.8. Table 10.1 on page 397 lists the most useful arguments to `envelope`.

Envelopes computed in the manner of Figure 7.27 are called *pointwise* envelopes because the maximum and minimum values are calculated separately for each distance r (i.e. for each 'point' on the horizontal axis). A very important caveat about pointwise envelopes is that they can only be interpreted as a statistical significance test if the distance value r^* was chosen in advance. It is not valid to first generate the Figure and then choose a favourable value of r; this would be 'data snooping'. If we allow ourselves this freedom then we are much more likely to find a value of r where the empirical $K(r)$ lies outside the envelopes, so the result is much less significant. The same caveat is emphasised in [572, 431, 40] and is discussed at length in Chapter 10.

7.8.2 Global envelopes

The problem of 'data snooping' can be avoided using *global envelopes*, shown in Figure 7.28. As distinct from the pointwise envelopes in Figure 7.27, the global envelopes in Figure 7.28 delimit a zone of *constant width*. The width is determined by finding the most extreme deviation from the theoretical K-function that is achieved by any of the simulated K-functions, at any distance r along the horizontal axis.

That is, for each simulated dataset, we compute the maximum vertical deviation D between the graphs of $\widehat{K}$ and K_{pois} over some range of distances. The maximum D_{max} of the deviations for all simulated datasets is taken. The global envelopes are

$$\begin{aligned} E_-(r) &= K_{pois}(r) - D_{max} \\ E_+(r) &= K_{pois}(r) + D_{max}. \end{aligned} \tag{7.36}$$

If the graph of the estimated K-function for the data transgresses these limits, it is statistically significant with a p-value of $1/(m+1)$ where m is the number of simulated patterns. Taking $m = 19$ gives a test with significance level 0.05.

The left panel of Figure 7.28 shows global envelopes for the K-function computed by

```
 > E <- envelope(swp, Kest, nsim=19, rank=1, global=TRUE)
```

A more powerful test is obtained if we (approximately) stabilise the variance, by using the L-function in place of K, as shown in the right panel of Figure 7.28. This clearly rejects the null hypothesis of CSR at the 0.05 significance level. The two panels used the *same* set of simulated point patterns.

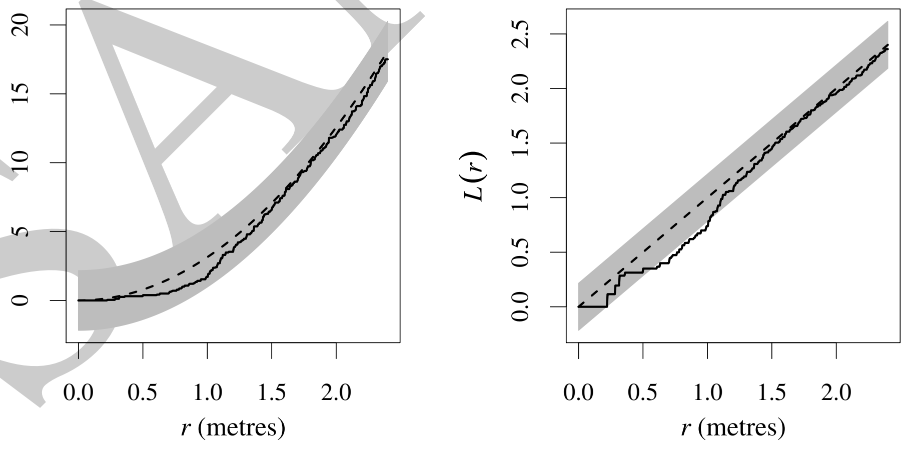

Figure 7.28. *Global envelopes relating to the Swedish Pines data.* Left: *global envelopes for the K-function.* Right: *global envelopes for the L-function. The same simulated patterns were used for both panels.*

Global envelopes were proposed by Ripley over 30 years ago [572, 574, 575], but they are still not widely used in applications, despite the fact that they provide a simple solution to the data snooping problem.

7.8.3 Non-graphical tests

Statistical tests can also be carried out without needing any graphical display. Two useful procedures are the MAD (maximum absolute deviation) test, and the DCLF (Diggle-Cressie-Loosmore-Ford) test, described in Sections 10.7.3 and 10.7.4, respectively. The MAD test [575] is equivalent to plotting the global envelopes described above and declaring a statistically significant outcome if the estimated *K*-function (etc.) wanders outside the global envelope anywhere:

```
> mad.test(swp, Lest, nsim=99, rmax=2, use.theo=TRUE)
        Maximum absolute deviation test of CSR
        Monte Carlo test based on 99 simulations
        Summary function: L(r)
        Reference function: theoretical
        Alternative: two.sided
        Interval of distance values: [0, 2] metres

data:  swp
mad = 0.29999, rank = 1, p-value = 0.01
```

The DCLF test [223, p. 122], [190, eq. (8.5.42), p. 667], [431] is based on the mean squared deviation between the empirical summary function and the theoretical function over a range of distance values. The DCLF test is more powerful than the MAD test provided the maximum distance is chosen carefully; the MAD test is more reliable when little information is available about the range of dependence [40].

```
> dclf.test(swp, Lest, nsim=99, rmax=2, use.theo=TRUE)$p.value
[1] 0.01
```

Both tests indicate there is strong evidence that the Swedish Pines data are not completely random.

7.9 Detecting anisotropy

A point process is 'isotropic' if all its statistical properties are unchanged when it is rotated (Section 5.6.3). If there is any statistical property of the process that does change when the process is rotated, then the process is 'anisotropic'. A point process could be stationary, but not isotropic. The microstructure of a mineral could be homogeneous, but with a preferential direction, perhaps reflecting the history of deformation [519, 355].

The following code generates a synthetic point pattern which is homogeneous and anisotropic. First we generate a Simple Sequential Inhibition process, and then deform space by squashing the *y*-axis.

```
> X <- rSSI(0.05, win=owin(c(0,1), c(0, 3)))
> Y <- affine(X, mat=diag(c(1, 1/3)))
```

Figure 7.29 shows the simulated pattern Y and its Fry plot, clearly indicating that the pattern is anisotropic.

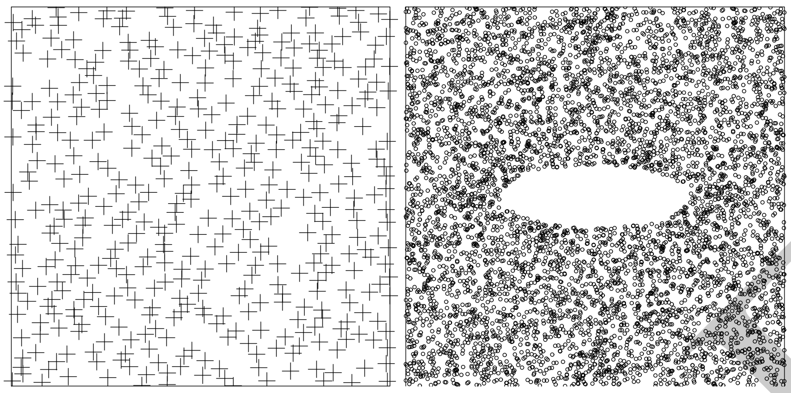

Figure 7.29. *Anisotropy.* Left: *synthetic point pattern exhibiting anisotropy.* Right: *Fry plot.*

7.9.1 Sector K-function

Various modifications of the K-function have been suggested [520, 636] for measuring anisotropy.

Instead of counting all data points falling inside a circle of radius r centred at a data point, we could replace the circle by another geometrical shape.

The *sector* shown in Figure 7.30 is that part of the disc of radius r lying between two lines at orientations α and β. Here r is the function argument, while the orientations α and β are fixed. The *sector K-function* of a stationary point process is $1/\lambda$ times the expected number of points lying within this sector, given that the vertex of the sector is a point of the process. The sector K-function can be estimated using the same edge-correction techniques as the original K-function.

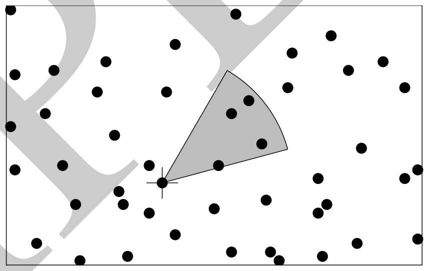

Figure 7.30. *Geometry of the sector K-function.*

The `spatstat` function `Ksector` computes the empirical sector K-function using the translation edge correction, together with the theoretical value for a Poisson process, $K_{sector,\, pois}(r) = (\beta - \alpha)r^2/2$ if α, β are expressed in radians.

For example, to compute the sector K-functions for two 30-degree angle sectors centred on the x and y axes, respectively:

```
 > Khoriz <- Ksector(Y, begin = -15, end = 15, units="degrees")
 > Kvert <- Ksector(Y,  begin = 90-15, end = 90+15, units="degrees")
```

Note that angles are measured anticlockwise from the x-axis. Anisotropy would be suggested if these two functions appeared to be unequal. We may then compare them by superimposing the plots, or by using `eval.fv` to compute the difference between the functions. The left panel of Figure 7.31 shows the result of

```
> plot(Khoriz, trans/theo ~ r, lty=2)
> plot(Kvert, trans/theo ~ r, add=TRUE)
```

where we have divided each estimate by the theoretical value.

The right panel of Figure 7.31 shows the difference `Khoriz-Kvert` and a 95% confidence interval computed by the following code:

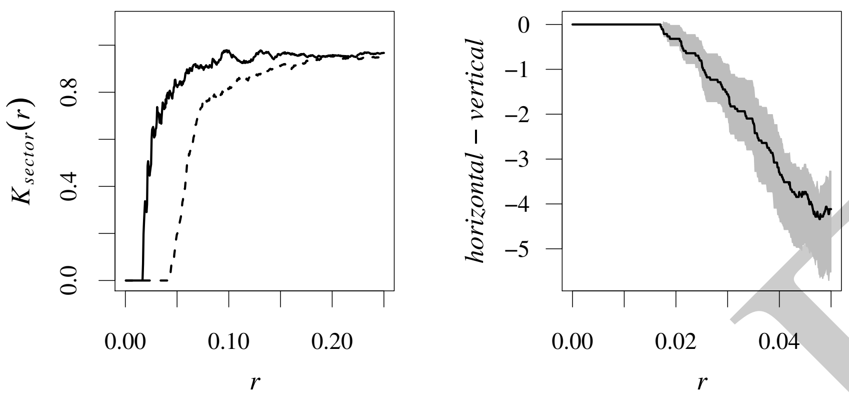

Figure 7.31. *Anisotropy analysis using* `Ksector` *for the synthetic pattern in Figure 7.29.* Left: *Superimposed plot of normalised* `Ksector` *functions for 30-degree sectors centred on the horizontal axis (dashed line) and vertical axis (solid line).* Right: *95% confidence interval for the difference ($\times 10^4$) between horizontal and vertical sector K-functions using block bootstrap.*

```
> dK <- function(X, ...) {
    K1 <- Ksector(X, ..., begin = -15, end = 15, units="degrees")
    K2 <- Ksector(X, ..., begin = 90-15, end = 90+15, units="degrees")
    eval.fv(K1-K2)
  }
> CIdK <- varblock(Y, dK, nx=5)
```

The sector K-function requires a choice of the limiting angles α and β, so is best used when there is some prior knowledge about the preferential directions in the pattern.

7.9.2 Pair orientation distribution

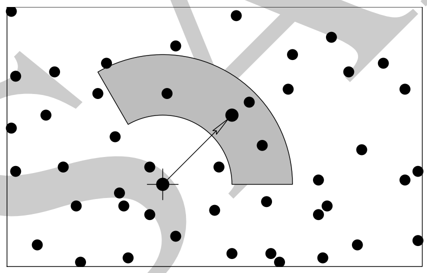

Figure 7.32. *Geometry for the point pair orientation distribution function.*

Anisotropy can also be measured by observing using the *directions* of arrows joining pairs of points.

Consider all pairs of points in a point pattern that lie more than r_1 and less than r_2 units apart, where $r_1 < r_2$ are fixed distances. For each such pair, we measure the direction of the arrow joining the points, as an angle in degrees anticlockwise from the x-axis. The probability distribution of these angles is the *point pair orientation distribution*.

Figure 7.32 shows the geometry for estimating the cumulative distribution function of the angles, called the 'point pair orientation distribution function' [638, (14.53), p. 271]

$$O_{r_1,r_2}(\phi) = \frac{\mathbb{E}\left[\text{number of points in } O(u,r_1,r_2,\phi) \,\middle|\, \mathbf{X} \text{ has a point at location } u\right]}{\mathbb{E}\left[\text{number of points in } O(u,r_1,r_2,360) \,\middle|\, \mathbf{X} \text{ has a point at location } u\right]}, \quad 0 \le \phi \le 360 \tag{7.37}$$

where $O(u,r_1,r_2,\phi)$ is the region bounded by circles of radius r_1 and r_2 centred at u and lines at orientation 0 and ϕ through u, shown in Figure 7.32. As Figure 7.32 suggests, this summary function can be estimated in the same way as the K-function.

The spatstat function pairorient computes an estimate of either the cumulative distribution function or the probability density of angles. The default is to compute the probability density, using a kernel smoothing estimate:

```
> f <- pairorient(Y, r1=0.02, r2=0.05, sigma=5)
```

The result is an "fv" object. Although this could be plotted by plot.fv, the display is challenging to interpret. For probability distributions of angles, it is usually better to display a rose diagram or *rose of directions* plot, in which the function value is interpreted as radial distance from the centre of the plot.

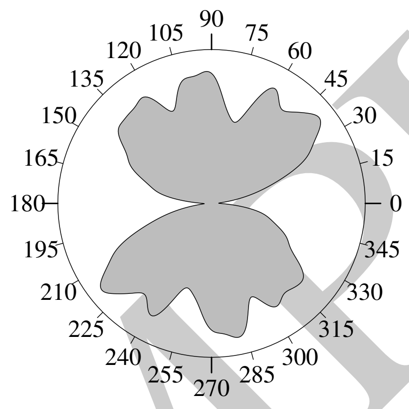

Figure 7.33. *Rose diagram of estimated probability density for the point pair orientation distribution of the synthetic pattern in Figure 7.29.*

Figure 7.33 shows a rose diagram for the estimated probability density f, generated by rose(f). It strongly indicates a preferential direction around 90 and 270 degrees. That is, at distances between 0.02 and 0.05, pairs of data points tend to be either directly above or directly below each other, and are rarely found directly to one side of each other.

The point pair orientation distribution can be an effective tool for finding a preferential direction, but seems to be quite sensitive to the choice of the distances r_1 and r_2. An alternative tool is the nearest-neighbour orientation distribution (Section 8.8).

7.9.3 Anisotropic pair correlation function

These considerations lead us back to the Fry plot (Section 7.2.2). In a stationary point process, virtually all information about the correlation structure is contained in the Fry plot. For example, the K-function could be calculated from the Fry plot, by counting how many dots in the Fry plot fall in a circle of radius r centred at the origin, and standardising appropriately. Replacing the circle by another 'test set', such as a sector or a ring, gives the other summary functions described in this chapter.

By counting dots in the Fry plot we are effectively treating the Fry plot as a point process in its own right, and finding the *intensity* of this process. The intensity measure of the Fry plot, appropriately standardised, is called the *reduced second moment measure* of the original point process. It generalises Ripley's K-function from circles to test sets of general shape [674, 520, 638].

Definition 7.2. *For a stationary point process* $\mathbf{X}$ *with intensity* λ*, the* ***reduced second moment***

measure *is the measure defined for any test set A in two-dimensional space by*

$$\mathscr{K}(A) = \frac{1}{\lambda}\mathbb{E}\left[\textit{number of points of } \mathbf{X} \textit{ (except u) falling in } A+u \,\middle|\, \mathbf{X} \textit{ has a point at } u\right] \tag{7.38}$$

for an arbitrary location u.

For example, if we take the test set $A = b(0,r)$, the disc of radius r centred at the origin, then $A+u$ is the disc of radius r centred at u, and equation (7.38) reduces to (7.4), so that $\mathscr{K}(A) = K(r)$, the value of Ripley's K-function for distance r. See Section 7.13 for details.

The measure $\mathscr{K}$ can be regarded as the generalisation of the K-function, and also as a standardised version of the intensity measure of the Fry plot. The connection between (7.38) and the Fry plot is that a point x_j in $\mathbf{X}$ falls in $A+u$ if and only if the vector difference $x_j - u$ falls in A.

Be warned that some writers use the term *reduced second moment measure* when they mean the K-function. This has caused confusion. As originally defined [457, 197], the reduced second moment measure is a measure, while the K-function is a function obtained by evaluating this measure for discs of increasing radius.

In most applications we can assume that the Fry plot has an intensity function, so that $\mathscr{K}$ has an intensity or density function, the *anisotropic pair correlation function* $g(v)$ defined for vectors v. This can be estimated by kernel smoothing *of the Fry plot* with edge correction and standardisation. The left panel of Figure 7.34 shows an estimate of the anisotropic pair correlation function for the point pattern data in Figure 7.29.

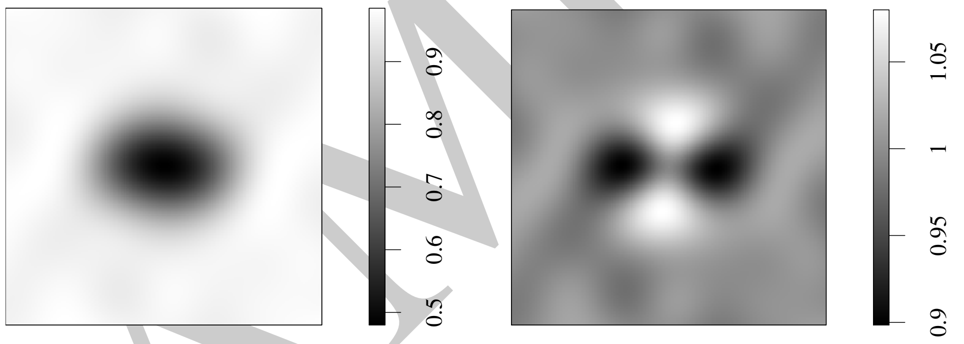

Figure 7.34. *Anisotropic pair correlation function.* Left: *image of estimated anisotropic pair correlation $g(u)$ for the synthetic pattern in Figure 7.29, as a function of vector difference u, with origin at centre of image. Detail of function inside the square of side length 0.2 centred at the origin.* Right: *anisotropy ratio $r(u) = g(u)/\overline{g}(\|u\|)$ where $\overline{g}$ is the rotational mean of g.*

The anisotropic pair correlation function $g(v)$ is a generalisation of the pair correlation function $g(r)$ to anisotropic, stationary point processes. It allows the correlation between pairs of points to be dependent on their relative position (the vector v) and not only on the distance r between them. Again, $g(v)$ is standardised so that it is equal to 1 when the process is completely random.

The anisotropic pair correlation function can be interpreted as a joint probability, as in Figure 7.22 on page 226. Imagine the observation window is divided into a fine grid of pixels, and consider two pixels u and v separated by the vector $v-u$. If $p_2(u,v)$ is the probability that *both* pixels contain random points, then

$$g(v-u) \doteq \frac{p_2(u,v)}{p^2} \tag{7.39}$$

where p^2 is the corresponding probability for a Poisson process.

The spatstat package provides a function Kmeasure which computes an estimate of the anisotropic pair correlation function (the density of the reduced second moment measure $\mathscr{K}$), effectively by smoothing the Fry plot. The algorithm approximates the point pattern and its window by pixel images, introduces a Gaussian smoothing kernel, and uses the Fast Fourier Transform to compute a density estimate of $\mathscr{K}$, effectively using the translation edge correction. The standard deviation σ of the kernel must be chosen by hand; there do not seem to be any established rules for bandwidth selection in this context.

The left panel of Figure 7.34 shows the estimated anisotropic pair correlation function $g(v)$ as a function of the two-dimensional vector v, with the origin in the centre of the picture. This is effectively a smoothed and renormalised version of the Fry plot. It was generated by the commands

```
> ganiso <- Kmeasure(Y, sigma=0.02, eps=0.001)
> detail <- square(c(-0.1,0.1))
> plot(ganiso[detail])
```

Choosing the bandwidth sigma=0.02 and pixel resolution eps=0.001 ensures that we do not oversmooth or pixellate interesting detail, which (according to the Fry plot) is at a scale of about 0.1 units.

Evidence for anisotropy can be judged by comparing the value of $g(v)$ for different vectors v of the same length but different orientations. A useful tool is to compute the *rotational mean*

$$\overline{g}(t) = \frac{1}{2\pi}\int_0^{2\pi} g((t\cos\theta, t\sin\theta))\,\mathrm{d}\theta \tag{7.40}$$

which gives, for each distance t, the average value of $g(v)$ over all vectors v of length $\|v\| = t$. Then the *anisotropy ratio* $r(v) = g(v)/\overline{g}(\|v\|)$ is a comparison of the anisotropic pair correlation with its rotational mean.

The spatstat function rotmean computes the rotational mean of any image. The right panel of Figure 7.34 shows the result of plotting the anisotropy ratio $g(v)/\overline{g}(\|v\|)$, computed by

```
> giso <- rotmean(ganiso, result="im")
> grel <- ganiso/giso
> plot(grel[detail])
```

The right panel of Figure 7.34 gives a clear impression of anisotropy, with anisotropy ratios above 1 in the vertical direction and below 1 in the horizontal direction, at distances of about 0.04 units. The left panel shows that this corresponds to pair correlation values closer to 1 in the vertical direction, and further away from 1 in the horizontal direction, at this distance range. This is consistent with the appearance of the Fry plot.

For a stationary isotropic point process there is a close relationship between the K-function and the isotropic pair correlation function through equations (7.22) and (7.25). Equation (7.25) can be generalised to a stationary, *anisotropic* point process:

$$\mathscr{K}(A) = \int_A g(v)\,\mathrm{d}v. \tag{7.41}$$

In particular, for $A = b(0,r)$,

$$K(r) = \int_{b(0,r)} g(v)\,\mathrm{d}v. \tag{7.42}$$

Equation (7.41) can be used to compute an estimate of the second moment measure $\mathscr{K}(A)$ for any region A. We first use Kmeasure to compute an estimate of g as a pixel image, then compute the integral of the pixel image over the domain A using integral.im. For example, if A is the square window detail created above,

```
> integral(ganiso, detail)
[1] 0.03681638
```

The pair correlation function also has a direct relationship to the second moment of the number of points falling in a region B. If $N = n(\mathbf{X} \cap B)$ then $N(N-1)$ is the number of ordered pairs of distinct points falling in B; the expected number of such pairs is

$$\mathbb{E}[N(N-1)] = \lambda^2 \int_B \int_B g(v-u)\,\mathrm{d}u\,\mathrm{d}v \tag{7.43}$$

as we can see intuitively by dividing B into a fine grid of pixels, and summing over all pairs of distinct pixels the joint probability that both pixels contain random points.

The result above is a special case of Campbell's formula for *second* moments. Consider the sum, over all pairs of distinct random points x_i, x_j where $i \neq j$, of some function $f(x_i, x_j)$. The second moment version of Campbell's formula states that

$$\mathbb{E}\left[\sum_{x_i \in \mathbf{X}} \sum_{\substack{x_j \in \mathbf{X} \\ x_j \neq x_i}} f(x_i, x_j)\right] = \lambda^2 \int \int f(u,v) g(v-u)\,\mathrm{d}u\,\mathrm{d}v \tag{7.44}$$

provided the expectation is finite. This is the analogue of equation (6.11) on page 169, for second moments.

7.10 Adjusting for inhomogeneity

If a point pattern is known or suspected to be spatially inhomogeneous, then our statistical analysis of the pattern should take account of this inhomogeneity. For general discussion see [355, sec. 4.10], [230, 225].

7.10.1 The general pair correlation function

First we need to extend the concept of the pair correlation function to point processes which cannot be assumed to be stationary or isotropic.

Suppose the point process $\mathbf{X}$ has intensity function $\lambda(u)$, so that

$$\mathbb{E}n(\mathbf{X} \cap A) = \int_A \lambda(u)\,\mathrm{d}u$$

for any bounded region A. Additionally suppose that $\mathbf{X}$ has **second moment intensity function** $\lambda_2(u,v)$, also called the **product density**, defined as the function that satisfies

$$\mathbb{E}[n(\mathbf{X} \cap A)\, n(\mathbf{X} \cap B)] = \int_A \int_B \lambda_2(u,v)\,\mathrm{d}u\,\mathrm{d}v \tag{7.45}$$

for any disjoint regions A and B. For a Poisson process with intensity function $\lambda(u)$, we have $\lambda_2(u,v) = \lambda(u)\lambda(v)$.

The second moment intensity function has a simple interpretation. Again imagine that the observation window is divided into a fine grid of pixels of area a. Let $p(u)$ be the probability that the pixel with centre u contains a random point. Then $p(u) \doteq \lambda(u)a$. Let $p_2(u,v)$ be the joint probability that the two pixels with centres u and v both contain random points. Then $p_2(u,v) \doteq \lambda_2(u,v)a^2$.

In this general setting we can define the general *pair correlation function*

$$g_2(u,v) = \frac{\lambda_2(u,v)}{\lambda(u)\lambda(v)} \tag{7.46}$$

for any two different spatial locations u and v. Using the approximations above, $g_2(u,v)$ is approximately $p_2(u,v)/(p(u)p(v))$, the probability that pixels centred at u and v both contain random points, divided by the corresponding probability for a Poisson process with the same intensity function $\lambda(u)$.

7.10.2 Inhomogeneous K and g functions

Estimation of the function $g_2(u,v)$ is not practically possible without some further assumption on the form of the function. A strategy proposed in [46] effectively assumes that the point process is *correlation-stationary*[4]

$$g_2(u,v) = g(v-u), \tag{7.47}$$

that is, the pair correlation between u and v depends only on their relative position. This would be true if the process is stationary, but is also true for an inhomogeneous Poisson process, and for many other processes [678].

Assuming (7.47), it is possible to define a counterpart of $K(r)$ called the *inhomogeneous K-function*. The idea is that each point x_i will be weighted by $w_i = 1/\lambda(x_i)$, the reciprocal of the intensity at x_i, and each pair of points x_i, x_j will be weighted by $w_{ij} = w_i w_j = 1/(\lambda(x_i)\lambda(x_j))$.

The *inhomogeneous K-function* is defined as

$$K_{inhom}(r) = \mathbb{E}\left[\sum_{x_j \in \mathbf{X}} \frac{1}{\lambda(x_j)} \mathbf{1}\left\{0 < ||u - x_j|| \le r\right\} \;\middle|\; u \in \mathbf{X} \right] \tag{7.48}$$

assuming this does not depend on location u. If (7.47) holds, then (7.48) does not depend on u.

Thus, $K_{inhom}(r)$ is the expected total 'weight' of all random points within a distance r of the point u, where the 'weight' of a point x_i is $1/\lambda(x_i)$. If the process is stationary, then $\lambda(u)$ is constant and $K_{inhom}(r)$ reduces to the usual K-function (7.6).

For an inhomogeneous Poisson process with intensity function $\lambda(u)$, the inhomogeneous K-function is

$$K_{inhom,\ pois}(r) = \pi r^2 \tag{7.49}$$

exactly as for the homogeneous case.

The standard estimators of K can be extended to the inhomogeneous K-function:

$$\widehat{K}_{inhom}(r) = \frac{1}{D^p |W|} \sum_i \sum_{j \ne i} \frac{\mathbf{1}\left\{||x_i - x_j|| \le r\right\}}{\widehat{\lambda}(x_i)\widehat{\lambda}(x_j)} e(x_i, x_j; r) \tag{7.50}$$

where $e(u,v,r)$ is an edge correction weight as before, and $\widehat{\lambda}(u)$ is an estimate of the intensity function $\lambda(u)$. The constant D^p in (7.50) is the pth power of

$$D = \frac{1}{|W|} \sum_i \frac{1}{\widehat{\lambda}(x_i)} \tag{7.51}$$

which has expected value 1 if the intensity is estimated without error. Choosing the power p equal to 1 or 2 improves statistical performance. Theoretical results in [46] show that, if the intensity function is *known*, or can be estimated with very high accuracy, then $\widehat{K}_{inhom}(r)$ is an unbiased estimator of $K_{inhom}(r)$ when $p = 0$, and is approximately unbiased when $p = 1$ or 2.

[4]Called 'second order intensity-reweighted stationary' (soirs) in [46].

In practice, the intensity function will need to be estimated from data, so the estimator (7.50) could be biased. The intensity function is usually estimated from the *same* point pattern data, which can lead to substantial bias in the estimate of $K_{inhom}(r)$. This justifies the data-dependent denominator D. Nonparametric estimates of intensity, being more responsive to local fluctuations in the data, tend to produce more biased estimates of $K_{inhom}(r)$ than parametric estimates of intensity, provided the intensity model is correctly specified [46, 225, 274].

The estimator (7.50) requires the estimated intensities $\widehat{\lambda}(x_i)$ at the data points x_i. To reduce bias it is advisable to use a *leave-one-out* intensity estimator as described in Section 6.5.1.3.

The empirical inhomogeneous K-function is computed in `spatstat` by the command `Kinhom(X, lambda)` where `X` is the point pattern and `lambda` is the estimated intensity function. Here `lambda` may be a pixel image, a `function(x,y)` in the R language, a fitted point process model, a numeric vector giving the values $\widehat{\lambda}(x_i)$ at the data points x_i only, or it may be omitted (and will then be estimated from `X`). By default, the data-dependent denominator D^1 is used: the power p is controlled by the argument `normpower`.

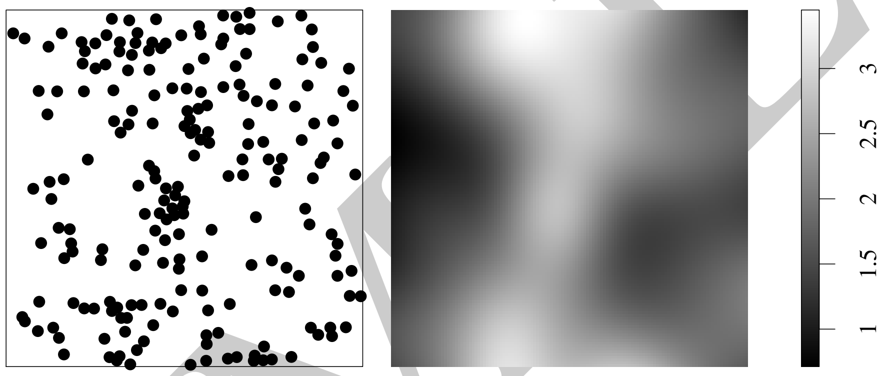

Figure 7.35. *Full dataset of Japanese Black Pines of Numata and Ogata (*Left*) and kernel-smoothed intensity estimate (*Right*). Smoothing bandwidth selected by likelihood cross-validation.*

Figure 7.35 shows the full dataset of Japanese Black Pine seedlings and saplings in a 10 metre square study region recorded by Numata [503] and extensively studied by Ogata [514, 516]. A kernel-smoothed intensity estimate is shown, with smoothing bandwidth selected by likelihood cross-validation using `bw.ppl` (see Section 6.5.1.2 on page 170):

```
> numata <- residualspaper$Fig1
> lambda <- density(numata, bw.ppl)
```

The chosen bandwidth seems appropriate, and the intensity estimate gives a strong impression of inhomogeneity. We now compute the inhomogeneous K-function:

```
> numataK <- Kinhom(numata, lambda)
```

The result is shown in Figure 7.36. The edge-corrected estimates are close to the Poisson theoretical value, suggesting that the data are consistent with an inhomogeneous Poisson process with the intensity shown in Figure 7.35.

If the intensity function is not given in the call to `Kinhom` then it will be estimated by kernel smoothing using the leave-one-out estimator. We could have obtained the same result by

```
> numataK <- Kinhom(numata, sigma=bw.ppl)
```

The argument `sigma=bw.ppl` is passed to `density.ppp`. In practice the results obtained from this procedure can be quite sensitive to the choice of bandwidth.

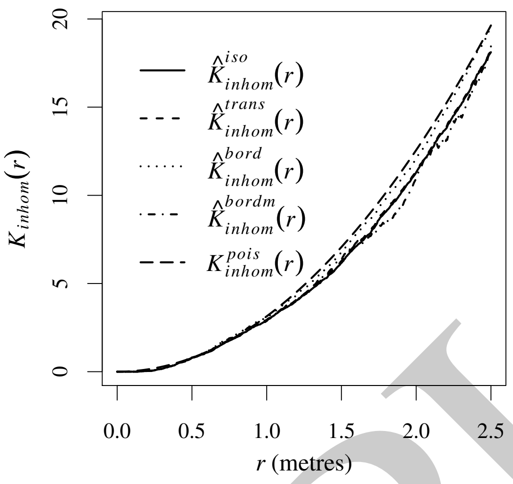

Figure 7.36. *Inhomogeneous K-function for full Japanese Pines data, using the estimated intensity function from the right panel of Figure 7.35.*

The inhomogeneous analogue of the *L*-function is

$$L_{inhom}(r) = \sqrt{K_{inhom}(r)/\pi}. \tag{7.52}$$

For an inhomogeneous Poisson process, $L_{inhom,\ pois}(r) \equiv r$. The corresponding empirical *L*-function is again justified by the fact that it approximately stabilises the variance. The inhomogeneous *L*-function can be computed in `spatstat` using `Linhom`, which has the same arguments as `Kinhom`.

The relationship (7.42) between the *K*-function and pair correlation function of a stationary point process extends to the inhomogeneous *K*-function:

$$K_{inhom}(r) = \int_{b(0,r)} g_2(0,v)\,\mathrm{d}v. \tag{7.53}$$

In general $g_2(0,v)$ depends on the orientation of v as well as the distance $\|v\|$. The rotational mean of the pair correlation

$$\overline{g}(r) = \frac{1}{2\pi}\int_0^{2\pi} g_2(0,(r\cos\theta, r\sin\theta))\,\mathrm{d}\theta$$

is the 'inhomogeneous pair correlation function' $g_{inhom}(r)$, and satisfies

$$g_{inhom}(r) = \frac{K'_{inhom}(r)}{2\pi r}.$$

The inhomogeneous pair correlation function is computed by `pcfinhom`:

```
> g <- pcfinhom(bei)
```

The previously mentioned method `pcf.fv`, which converts a *K*-function estimate into a pair correlation function estimate by numerical differentiation, also works for estimates of the inhomogeneous *K*-function. Thus the following is an alternative to the foregoing estimation procedure:

```
> g <- pcf(Kinhom(bei))
```

To construct confidence intervals for the true value of $K_{inhom}(r)$ or $g_{inhom}(r)$ one can use `varblock` or `lohboot` as described in Section 7.7.

To test the hypothesis that the point process is an inhomogeneous Poisson process, use `envelope`, `dclf.test`, or `mad.test`. Some care is required: as explained in Chapter 10, it is important to ensure that the simulated patterns are treated in exactly the same way as the original data pattern was treated. Figure 7.37 shows the global envelopes of the centred inhomogeneous L function for the full Japanese Pines data of Figure 7.35, generated by

```
> lam <- density(numata, bw.ppl)
> E <- envelope(numata, Linhom, sigma=bw.ppl,
                simulate=expression(rpoispp(lam)),
                use.theory=TRUE, nsim=19, global=TRUE)
> plot(E, . - r ~ r)
```

The plot indicates significant evidence against the inhomogeneous Poisson model, with a suggestion that there is inhibition at small distances.

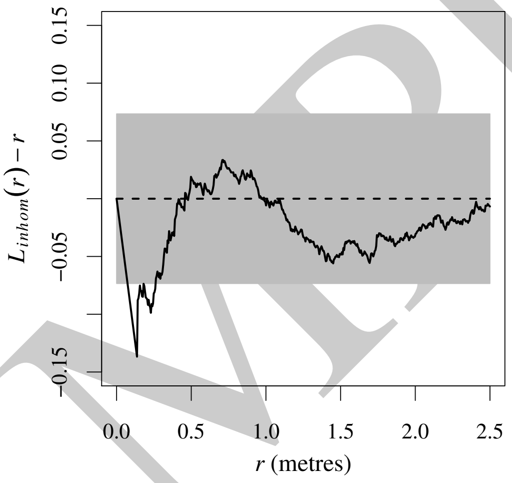

Figure 7.37. *Centred inhomogeneous L-function $L_{inhom}(r) - r$ for the Numata-Ogata Japanese pines data (black line) and global envelopes from 19 simulated realisations of an inhomogeneous Poisson process with intensity estimated by a leave-one-out kernel smoother.*

We emphasise that — despite its name — the inhomogeneous K-function does not apply to every spatially inhomogeneous point process; it applies only when the point process is correlation-stationary, equation (7.47). Substantial differences between the estimates of the inhomogeneous K-function obtained from different subsets of the data, or obtained using different edge corrections, would suggest that this assumption is false. A formal hypothesis test for correlation-stationarity is described in Section 16.8.5.

7.10.3 Local scaling

The inhomogeneous K-function effectively assumes that the *spatial scale* of interaction remains constant, while the intensity is spatially varying. This is not an appropriate assumption for the bronze filter data in Figure 1.9 on page 8, an inhomogeneous point pattern in which the spacing between points increases gradually from left to right.

An alternative approach to inhomogeneity [312, 556, 309, 311] is to assume that the point process is equivalent, in small regions, to a rescaled version of a 'template' process, where the template process is stationary and isotropic, and the rescaling factor can vary from place to place. This could be an appropriate model for Figure 1.9.

We would then be assuming that, for two locations u and v sufficiently close together,

$$g(u,v) = g_1\left(\frac{\|u-v\|}{s}\right) \tag{7.54}$$

where g_1 is the pair correlation function of the 'template' process, and s is the local scaling factor applicable to both locations u and v. Rescaling the spatial coordinates by a factor $1/s$ rescales the intensity by s^2, so the appropriate value is $s = 1/\sqrt{\lambda}$ where λ is the local intensity in the neighbourhood of u and v.

In practice, we would first estimate the intensity function of the original data by $\widehat{\lambda}(u)$, then for each pair of data points x_i, x_j define the rescaled distance

$$d^*_{ij} = \frac{\|x_i - x_j\|}{s(x_i,x_j)}$$

where the rescaling factor is

$$s(x_i,x_j) = \frac{1}{2}\left(\frac{1}{\sqrt{\widehat{\lambda}(x_i)}} + \frac{1}{\sqrt{\widehat{\lambda}(x_j)}}\right).$$

An edge-corrected estimator of Ripley's original K-function is then applied to the distances d^*_{ij} to give the *locally scaled K-function* [312, 556, 311].

The `spatstat` package provides the commands `Kscaled` and `Lscaled` which compute the locally scaled K- and L-functions. Their syntax is similar to `Kinhom`.

To estimate a locally scaled K-function for the bronze filter data (Figure 1.9), we first estimated the intensity, assuming it is an exponential function of the x-coordinate, by fitting a point process model (see Chapter 9):

```
> X <- unmark(bronzefilter)
> fit <- ppm(X ~ x)
> lam <- predict(fit)
```

The locally scaled K-function was then estimated by

```
> Kbro <- Kscaled(X, lam)
```

The result is plotted in Figure 7.38.

Again we emphasise that the locally scaled K-function is applicable only when the point process is locally scaled. A formal hypothesis test for local scaling is described in Section 16.8.5.

7.11 Local indicators of spatial association

Another tool for handling inhomogeneous patterns is a LISA (Local Indicator of Spatial Association), obtained by decomposing a summary statistic into contributions from each of the data points.

For example, the empirical K-function (7.3) can be broken into contributions $\widehat{K}(r,x_i)$ from each individual data point x_i, shown in (7.33), known as the *local K-functions*. The usual estimate $\widehat{K}(r)$ is the average of the estimates $\widehat{K}(r,x_i)$ over all $i = 1,\dots,n$.

If it is suspected that the pattern may be a patchwork of different textures in different places, or

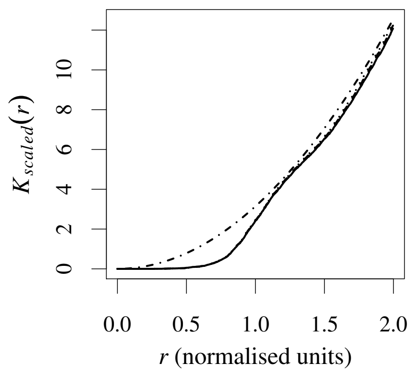

Figure 7.38. *Locally scaled K-function of the bronze filter data.*

that the pattern contains some anomalous features, then it can be useful to classify the n local K-functions $\widehat{K}(r,x_1),\ldots,\widehat{K}(r,x_n)$ into several groups of functions, perhaps using hierarhical clustering techniques. This approach was proposed independently in [281, 20] and developed in [186, 185]. Repeated measures analysis or functional principal component analysis [562] could be used to identify the main differences between the local functions.

The `spatstat` functions `localK`, `localL`, `localpcf` compute local versions of the K-function, L-function, and pair correlation function, respectively. Of course the L-function is not a *sum* of contributions from individual data points: the local L-function is arbitrarily defined as $\widehat{L}(r,x_i) = (\widehat{K}(r,x_i)/\pi)^{1/2}$. There are also *inhomogeneous* counterparts `localKinhom`, `localLinhom`, and `localpcfinhom` which are the contributions to the inhomogeneous K-function, etc, from each data point.

Local K-functions of the Swedish Pines data are computed by

```
> lK <- localK(swedishpines)
```

The result is an `"fv"` object with $n+2=73$ columns, where $n=71$ is the number of data points. The first n columns give the local K-functions $\widehat{K}(r,x_i)$ for each data point, in order. The final two columns contain the distance values r and the K-function for a Poisson process. Extracting only the function values by

```
> locK <- as.data.frame(lK)[, fvnames(lK, ".a")]
> rr   <- with(lK, r)
```

gives the data in a format ready for cluster analysis or functional data analysis. Normally `locK` would be transposed to `t(locK)` so that each row of the matrix contains one local K-function. Hierarchical clustering of the local K-functions, based on the mean square distance between functions, is implemented by

```
> locH <- hclust(dist(t(locK)))
```

A plot of `locH` would show a dendrogram of the local K-functions. See [186, 185] for further discussion.

The average of the local K-functions for the data points falling in a region A can also be computed by `Kest` using the argument `domain`.

7.12 Third- and higher-order summary statistics

We saw on page 211 that two different point processes can have identical first-order (intensity) and second-order (correlation) properties. In order to distinguish between such processes we need another approach. One possible strategy is to estimate the *third-order* moments, which would involve counting *triples* rather than *pairs* of data points.

For a stationary point process $\mathbf{X}$ with intensity λ, the triangle-counting function $T(r)$ is defined as the normalised expected number of triangles formed by points of $\mathbf{X}$, with all side lengths less than or equal to r, with one corner at a specified point of the process:

$$T(r) = \frac{1}{\lambda^3}\mathbb{E}\left[\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} m(x_i, x_j, u) \mid u \in \mathbf{X}\right] \tag{7.55}$$

where $m(a,b,c)$ is the maximum side length of the triangle with vertices a,b,c,

$$m(a,b,c) = \max\{\|a-b\|,\ \|a-c\|,\ \|b-c\|\}.$$

For a homogeneous Poisson process with intensity λ we have

$$T_{pois}(r) = \frac{\pi}{2}(\pi - \frac{3\sqrt{3}}{4})r^4. \tag{7.56}$$

Formal theory and edge corrections are given in [600].

Estimates of the triangle-counting function $T(r)$ are computed in `spatstat` by `Tstat`. For the point pattern `Xcell` shown in Figure 7.11, the empirical estimate $\widehat{T}(r)$ computed by `Tstat(Xcell)` is shown in Figure 7.39. This shows clear deviation from the values expected for a Poisson process. Simulation envelopes can be computed in the usual way.

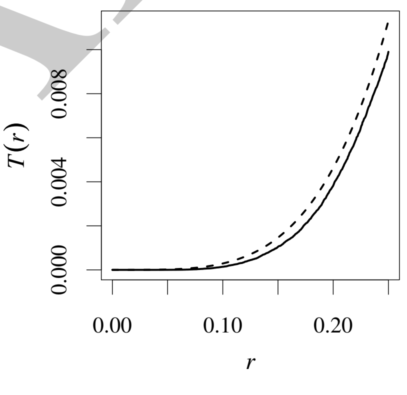

Figure 7.39. *Triangle-counting function $\widehat{T}(r)$ for the simulated realisation of the cell process in Figure 7.11. Solid line: translation correction estimate. Dashed line: theoretical value for Poisson process.*

Three-point correlation functions are frequently used in astronomy [537] and even *n*-point correlation functions are used [378, sec. 3.2], [445], [355, pp. 4.4.2].

7.13 Theory*

This section covers some basic theory for second moments of point processes. For more detail, see [484, Appendix C], [355, sec. 1.5], [27, 32, 569], [576, chap. 3]. For a full account, see [198, 199].

7.13.1 Second moment measures

Let $\mathbf{X}$ be a point process. We are interested in the variance of the count $n(\mathbf{X} \cap B)$,

$$\operatorname{var} n(\mathbf{X} \cap B) = \mathbb{E}[n(\mathbf{X} \cap B)^2] - [\mathbb{E} n(\mathbf{X} \cap B)]^2$$

and the covariance of two such counts,

$$\operatorname{cov}[n(\mathbf{X} \cap B_1), n(\mathbf{X} \cap B_2)] = \mathbb{E}[n(\mathbf{X} \cap B_1) n(\mathbf{X} \cap B_2)] - [\mathbb{E} n(\mathbf{X} \cap B_1)] [\mathbb{E} n(\mathbf{X} \cap B_2)].$$

A key observation is that $n(\mathbf{X} \cap B_1) n(\mathbf{X} \cap B_2)$ is equal to the number of ordered pairs (x, x') of points in the process $\mathbf{X}$ such that $x \in B_1$ and $x' \in B_2$.

Definition 7.3. *Let $\mathbf{X}$ be a point process in $\mathbb{R}^2$. Then $\mathbf{X} \times \mathbf{X}$ is a point process on $\mathbb{R}^2 \times \mathbb{R}^2$ consisting of all ordered pairs (x, x') of points $x, x' \in \mathbf{X}$. The intensity measure ν_2 of $\mathbf{X} \times \mathbf{X}$ is a measure on $\mathbb{R}^2 \times \mathbb{R}^2$ satisfying*

$$\nu_2(A \times B) = \mathbb{E}[N_{\mathbf{X}}(A) N_{\mathbf{X}}(B)].$$

This measure ν_2 is called the second moment measure *of* $\mathbf{X}$.

The second moment measure contains all information about the variances and covariances of the variables $N_{\mathbf{X}}(A)$. Campbell's formula (6.11) applied to $\mathbf{X} \times \mathbf{X}$ becomes

$$\mathbb{E}[\sum_{x_i \in \mathbf{X}} \sum_{x_j \in \mathbf{X}} f(x_i, y_j)] = \int_{\mathbb{R}^2} \int_{\mathbb{R}^2} f(u, v) \, \mathrm{d}\nu_2(u, v)$$

for a measurable function $f : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathsf{R}$.

For example, for the uniform Poisson point process of intensity $\lambda > 0$ in R^d, the second moment measure satisfies $\nu_2(A \times B) = \lambda^2 |A| \, |B| + \lambda |A \cap B|$. To simplify the calculation of certain moments, we introduce the *second factorial moment measure*

$$\nu_{[2]}(A \times B) = \mathbb{E}[n(\mathbf{X} \cap A) n(\mathbf{X} \cap B)] - \mathbb{E}[n(\mathbf{X} \cap A \cap B)].$$

This is the intensity measure of the process $\mathbf{X} * \mathbf{X}$ of all ordered pairs of *distinct* points of $\mathbf{X}$. It satisfies

$$\mathbb{E}[\sum_{x_i \in \mathbf{X}} \sum_{x_j \in \mathbf{X},\, j \neq i} f(x_i, x_j)] = \int_{\mathbb{R}^2} \int_{\mathbb{R}^2} f(u, v) \, \mathrm{d}\nu_{[2]}(u, v).$$

The name 'factorial' is derived from

$$\nu_{[2]}(A \times A) = \mathbb{E}[n(\mathbf{X} \cap A)^2] - \mathbb{E}[n(\mathbf{X} \cap A)] = \mathbb{E}[n(\mathbf{X} \cap A)[n(\mathbf{X} \cap A) - 1]].$$

For example, for the uniform Poisson process of intensity λ, the second factorial moment measure is $\nu_{[2]}(A \times B) = \lambda^2 |A| |B|$.

* Starred sections contain advanced material, and can be skipped by most readers.

Definition 7.4. *A point process* $\mathbf{X}$ *on* $\mathbb{R}^2$ *is said to have* second moment intensity λ_2 *if*

$$\nu_{[2]}(C) = \int_C \lambda_2(u,v)\,\mathrm{d}u\,\mathrm{d}v \tag{7.57}$$

for any compact $C \subset \mathbb{R}^2 \times \mathbb{R}^2$.

Informally, $\lambda_2(u,v)$ gives the joint probability that there will be points of $\mathbf{X}$ in infinitesimal regions around two specified locations u and v:

$$\mathbb{P}\{N(\mathrm{d}u) > 0,\, N(\mathrm{d}v) > 0\} \doteq \lambda_2(u,v)\,\mathrm{d}u\,\mathrm{d}v.$$

For example, the uniform Poisson process has second moment density $\lambda_2(x,y) = \lambda^2$. The binomial process of n points in W has $\lambda_2(u,v) = n(n-1)/|W|^2$ if $x,y \in W$, and zero otherwise.

Definition 7.5. *Suppose* $\mathbf{X}$ *is a point process on* $\mathbb{R}^2$ *which has an intensity function* $\lambda(x)$ *and a second moment intensity* $\lambda_2(u,v)$. *Then we define the* pair correlation function *of* $\mathbf{X}$ *by*

$$g_2(u,v) = \frac{\lambda_2(u,v)}{\lambda(u)\lambda(v)}.$$

For a uniform Poisson process of intensity λ, we have $\lambda(u) \equiv \lambda$ and $\lambda_2(u,v) \equiv \lambda^2$, so that $g_2(u,v) \equiv 1$. For a binomial process of n points in a region W, we have $g_2(u,v) \equiv 1 - 1/n$.

7.13.2 Second moments for stationary processes

For a *stationary* point process in $\mathbb{R}^2$, there is a 'disintegration' of the second moment measure.

Theorem 7.1. *Let* $\mathbf{X}$ *be a stationary point process on* $\mathbb{R}^2$ *with intensity* λ. *Then there is a measure* $\mathscr{K}$ *on* $\mathbb{R}^2$ *such that, for a general integrand* f,

$$\mathbb{E}\left[\sum_{x_i \in \mathbf{X}} \sum_{x_j \in \mathbf{X},\, j \neq i} f(x_i, x_j)\right] = \lambda \int\int f(u, u+v)\,\mathrm{d}\mathscr{K}(v)\,\mathrm{d}u. \tag{7.58}$$

$\mathscr{K}$ *is called the* reduced second moment measure *of* $\mathbf{X}$.

To understand the measure $\mathscr{K}$, we notice that for $A, B \subset \mathbb{R}^2$

$$\lambda|A|\mathscr{K}(B) = \int\int 1_A(u) 1_B(v-u)\,\mathrm{d}\nu_{[2]}(u,v) = \mathbb{E}\left[\sum_{x_i \in \mathbf{X}} \sum_{x_j \in \mathbf{X},\, j \neq i} 1_A(u) 1_B(v-u)\right].$$

This may also be obtained directly from (7.58) by taking $f(u,v) = 1_A(u)1_B(v-u)$. Since $\lambda|A| = \mathbb{E}n(\mathbf{X} \cap A)$, we have

$$\mathscr{K}(B) = \frac{\mathbb{E}\sum_{x_i \in \mathbf{X} \cap A} n(\mathbf{X} \cap (B + x_i) \setminus x_i)}{\mathbb{E}n(\mathbf{X} \cap A)}. \tag{7.59}$$

The right-hand side of (7.59) may be interpreted as the average, over all points x_i of the process, of the number of other points x_j of the process such that $x_j - x_i \in B$.

For the uniform Poisson process, $\mathscr{K}(B) = \lambda|B|$.

Suppose $\mathbf{X}$ is a stationary process on $\mathbb{R}^2$ which has a second moment density function λ_2. Then by comparing (7.57) with (7.58) we can see that $\lambda_2(u,v)$ depends only on $v - u$, say $\lambda_2(u,v) = \lambda^2 g(v-u)$ for some function g, and we can write

$$\mathscr{K}(B) = \lambda \int_B g(u)\,\mathrm{d}u.$$

This is the fundamental relationship between the pair correlation and the second moment measure.

7.14 FAQ

- *Should I use Ripley's K-function, the inhomogeneous K-function, or the scaled K-function?*

  This depends on the assumptions that are appropriate for your data. Using the K-function assumes the point process has homogeneous intensity and is correlation-stationary. Using the inhomogeneous K-function assumes the process is correlation-stationary. Hypothesis tests for checking these assumptions are described in Section 16.8.5.

- *When I plot the estimated K-function of my data using the command `plot(Kest(X))`, what is the quantity on the horizontal axis, and what units is it expressed in?*

  The horizontal axis is distance between points. If the plot annotation does not indicate the units in which the distance is measured, then the point pattern dataset `X` did not include a specification of the units of length. This should be repaired using the function `unitname`.

- *When I plot the estimated K-function of my data using the command `plot(Kest(X))`, the horizontal axis is the distance in metres, but what is the quantity on the vertical axis, and what units is it expressed in?*

  The quantity on the vertical axis is the average number of neighbours of a typical point, *divided by* the intensity (average number of points per unit area) so that different patterns can be compared. It is measured in units of area (number of points divided by points-per-unit-area). The reference benchmark is that for a completely random process the value of $K(r)$ is the area of the disc of radius r.

- *When I plot the estimated K-function of my data using the command `plot(Kest(X))`, the scale marks on the y-axis are huge numbers like `1e9`. Is this wrong? I don't see anything like this in your book.*

  The values of $K(r)$ are areas, and should be of the same order as the area of the observation window, expressed in the units you are using for the spatial coordinates. If your window is 30 kilometres across and the coordinates are recorded in metres, the window area is about $30000^2 \approx 10^9$ square metres. To avoid numerical overflow, it would be wise to rescale the spatial coordinates, for example converting metres to kilometres.

- *When I plot the estimated K-function of my data using the command `plot(Kest(X))`, I don't understand the meaning of the different coloured curves.*

  These curves are different estimates of the K-function, using different 'edge correction' techniques. Usually one of the curves is the theoretical value of the K-function, $K(r) = \pi r^2$, corresponding to a completely random pattern.

  The accompanying legend (plotted unless `legend=FALSE`) gives a mathematical expression for each of the edge corrections, and shows the corresponding line colour and line type used to plot the estimate.

  The return value of `plot.fv` is a table giving the line colour, line type, keyword, mathematical expression, and long text description of each curve that was plotted. For further information about the different edge corrections, see `help(Kest)`.

  The estimates of $K(r)$ by different edge correction techniques should be roughly equal. If the curves for the isotropic correction (`iso`), translation correction (`trans`), and border correction (`border`) estimates are wildly different, this suggests that estimation of K is difficult for these data (e.g., because there are too few data points, or the window is very narrow, or there are data points very close to the edge of the window).

For information on how to modify the plot of the K-functions, see `help(plot.fv)` or the examples in `help(Kest)`.

- *I have computed the inhomogeneous K-function using* `Kinhom`*. The different edge-corrected estimates have very different values. Which correction should I use?*

  Discrepancies between the edge-corrected estimates of the inhomogeneous K-function computed by `Kinhom(X, lambda)` usually occur when the intensity estimate `lambda` is inaccurate, or when the data are not correlation-stationary. First check that the estimated intensity appears to be reasonably accurate. If so, try changing the arguments `normalise` and `normpower`. If large discrepancies still remain, check for data points very close to the window boundary, which are treated differently by the different edge corrections. If there are no such points, the tentative conclusion is that the assumption of a correlation-stationary process is violated. Consider performing a test of this assumption (Section 16.8.5).

- *I can't seem to control the range of r values in* `plot(Kest(X))`*. How can I control it? How is the default plotting range determined?*

  Use the argument `xlim` to control the range of the x-axis, as documented in `help(plot.fv)`. For example, `plot(Kest(X), xlim=c(0, 7))`.

  An object of class `"fv"` contains function values for a certain range of r values (the *'available range'*, which is usually the maximum possible range). However, the default behaviour of `plot.fv` is to plot the function values for a narrower range of r values (the *'recommended range'*) which usually contains the important detail in the function. Both the available range and recommended range are printed when the `"fv"` object is printed.

  Using the argument `xlim` when plotting the K-function will override the recommended range. However, obviously we cannot choose `xlim` to be wider than the *available* range of r values. To extend the available range, you would need to re-compute the K-function, specifying the argument `r`. This should be a vector of closely spaced values of r ranging from 0 to the desired maximum.

- *When plotting a summary function using* `plot.fv`*, how can I control the legend position, text size, and other parameters?*

  See the help file for `plot.fv`. To set the legend text to half normal size, for example, set `legendargs=list(cex=0.5)`.

  To move the legend *outside* the plot area, call `plot.fv` with `legend=FALSE`, save the result, and use the columns in this data frame in a subsequent call to the `legend` function.

- *I have plotted a summary function using a formula such as* `sqrt(./pi) ~ r`*. Is it possible to save the plotted curves as another summary function?*

  Yes, if the right-hand side of the formula is the function argument. Use `with.fv`, for example `with(Kest(cells), sqrt(./pi))`

- *I used the* `envelope` *command to compute envelopes of the L-function. The printout for the envelope object says that it contains 'pointwise envelopes'. What value of r was used for these?*

  'Pointwise' means that for *each* chosen value of the distance r, the range of simulated values of $\widehat{L}(r)$ has been calculated, and a test based on these values would be valid. Plotting the pointwise envelopes shows what the result of the test would have been if we had chosen any particular value of r. For further explanation, see Chapter 10.

- *Is it possible to combine the empirical K-functions from several point patterns into a common K-function?*

  Yes; see Section 16.8.1.

- *I have mapped the locations of bird nests on a small island. Do I need to use an edge correction for $K(r)$?*

  This is an example of the 'Small World Model' (page 145). Strictly speaking the K-function is not defined because the point process is not stationary. A sensible strategy is to use the empirical K-function *without* edge correction (`Kest` with `correction="none"`) and to base inference on simulations from the appropriate null hypothesis.