Journal of Statistical Software

January 2005, Volume 12, Issue 6. *http://www.jstatsoft.org/*

spatstat: An R Package for Analyzing Spatial Point Patterns

Adrian Baddeley
University of Western Australia

Rolf Turner
University of New Brunswick

Abstract

spatstat is a package for analyzing spatial point pattern data. Its functionality includes exploratory data analysis, model-fitting, and simulation. It is designed to handle realistic datasets, including inhomogeneous point patterns, spatial sampling regions of arbitrary shape, extra covariate data, and 'marks' attached to the points of the point pattern.

A unique feature of **spatstat** is its generic algorithm for fitting point process models to point pattern data. The interface to this algorithm is a function `ppm` that is strongly analogous to `lm` and `glm`.

This paper is a general description of **spatstat** and an introduction for new users.

Keywords: conditional intensity, edge corrections, exploratory data analysis, generalised linear models, inhomogeneous point patterns, marked point patterns, maximum pseudolikelihood, spatial clustering .

1. Introduction

spatstat is one of several packages in the R language for analysing point patterns in two dimensions. [1] This paper is a general description of **spatstat** and may serve as an introduction for new users. Subsequent papers will cover advanced use of the package Baddeley and Turner (2005b) and explain its design and implementation Baddeley and Turner (2005a).

A simple example of a point pattern dataset is shown in Figure 1. The points represent the locations of seedlings and saplings of the Californian giant redwood.

Point pattern data may be much more complicated than Figure 1 suggests. The spatial sampling region in which the points were recorded may have arbitrary irregular shape, instead of being a rectangle as in Figure 1. The points may carry additional data (*marks*). For example, we may have recorded the height or the species name of each tree. There may be

[1] Alternatives include **splancs** Rowlingson and Diggle (1993); Bivand (2001), **spatial** Ripley (2001); Venables and Ripley (1997), **ptproc** Peng (2003) and **SSLib** Harte (2003).

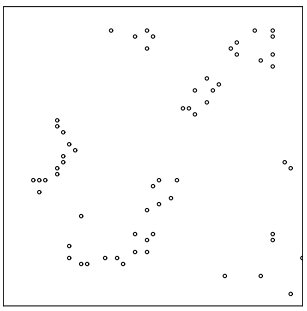

Figure 1: The classic Redwoods dataset Ripley (1977) available in **spatstat** as `redwood`.

additional covariate data which must be incorporated in the analysis. The **spatstat** package is designed to handle all these complications.

Figure 2 shows an example of a dataset which can be handled by **spatstat**; it consists of points of two types (plotted as two different symbols) and is observed within an irregular sampling region which has a hole in it. The label or 'mark' attached to each point may be a categorical variable, as in the Figure, or a continuous variable. See also Figures 6–9.

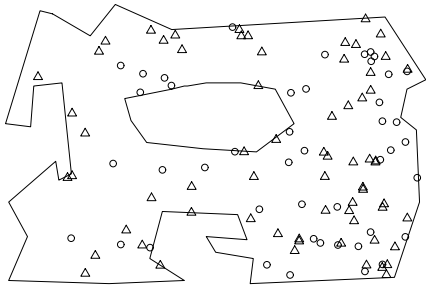

Figure 2: Artificial example demonstrating the complexity of datasets which **spatstat** can handle.

Point patterns analysed in **spatstat** may also be spatially inhomogeneous, and may exhibit dependence on covariates. The package can deal with a variety of covariate data structures. It will fit point process models which depend on the covariates in a general way, and can also simulate such models.

2. Goals

Our main reasons for writing **spatstat** were to:

Implement functionality. The research literature on spatial statistics provides a large body of techniques for analysing spatial point patterns (e.g. Bartlett (1975); Cliff and Ord (1981); Cressie (1991); Diggle (2003); van Lieshout (2000); Matérn (1986); Møller and Waagepetersen (2003); Moore (2001); Ripley (1981, 1988); Stoyan, Kendall, and Mecke (1995); Stoyan and Stoyan (1995); Upton and Fingleton (1985)). However, only a small fraction of these techniques have been implemented in software for general use.

Handle real datasets. New techniques published in the literature are often demonstrated only on a 'tame' example dataset, using a rudimentary proof-of-concept implementation. Such software is typically designed only for rectangular windows; the techniques themselves may assume that the point pattern is spatially homogeneous; and auxiliary information (such as covariate data) is often ignored.

For example, the classical redwood dataset of Figure 1 is a subset extracted by Ripley (1976, 1981) from a larger dataset of Strauss (1975) which is shown in Figure 3. The full dataset exhibits completely different spatial patterns on either side of the diagonal line shown on the plot. The diagonal line is a simple example of covariate data. As far as we are aware, the full dataset has never been subjected to comprehensive analysis.

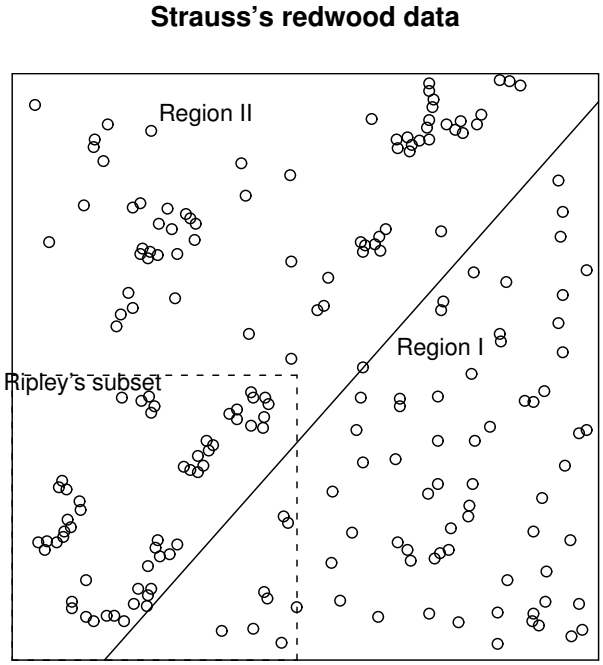

Figure 3: The full redwood dataset of Strauss (1975). The square in the bottom left corner shows the boundaries of the subset extracted by Ripley (1977) as the classical `redwood` dataset.

Similarly, Figure 4 shows the ant nest data of Harkness and Isham (1983). The full dataset records the locations of nests of two species of ants, observed in an irregular convex polygonal boundary, together with annotations showing a foot track through the region, and the boundary between field and scrub areas inside the region. Rectangular subsets of the data (marked "`A`" and "`B`" on the Figure) were analysed in Harkness and Isham (1983); Isham (1984); Takacs and Fiksel (1986); Högmander and Särkkä (1999); Baddeley and Turner (2000) and (Särkkä 1993, section 5.3). Again, as far as we are aware, the full dataset has never been subjected to detailed analysis inside the correct window.

Fit realistic models to data. In applications, the statistical analysis of spatial point patterns is conducted almost exclusively using 'exploratory' summary statistics such as the K function Cliff and Ord (1981); Cressie (1991); Diggle (2003); Møller and Waagepetersen (2003); Ripley (1988); Stoyan *et al.* (1995); Stoyan and Stoyan (1995); Upton and Fin-

gleton (1985). An important goal of **spatstat** is to fit parametric models to spatial point pattern data. Although methods for fitting point process models have been available since the 1970's Besag (1975); Diggle (2003); Ogata and Tanemura (1981, 1984); Møller and Waagepetersen (2003); Ripley (1981, 1988), most of these methods were very specific to the chosen model, and there were no software implementations of sufficient generality to fit realistic models to a real dataset. Recently we described an algorithm for fitting point process models of very general form Baddeley and Turner (2000). Our implementation of this algorithm has grown into the package **spatstat**.

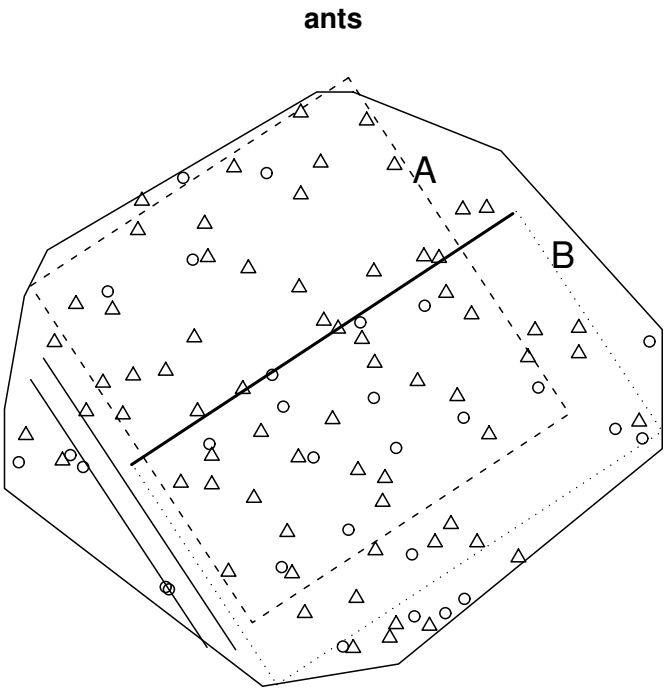

Figure 4: Harkness-Isham ant nests data. Map of the locations of nests of two species of ants, *Messor wasmanni* (△) and *Cataglyphis bicolor* (○) in an irregular region 425 feet in diameter. Data kindly supplied by Professors R.D. Harkness and V. Isham.

3. Capabilities

spatstat supports the following activities.

Creation, manipulation and plotting of point patterns: a point pattern dataset can easily be created, plotted, inspected, and transformed. Subsets of the pattern can easily be extracted (e.g. to thin the points or trim the window). Marks can readily be added or removed from a point pattern. Many geometrical transformations, operations and measurements are implemented.

Exploratory data analysis: standard empirical summaries of the data, such as the average intensity, the K function Ripley (1977) and the kernel-smoothed intensity map, can easily

be generated and displayed. Many other empirical statistics are implemented in the package, including the empty space function F, nearest neighbour distance function G, pair correlation function g, inhomogeneous K function Baddeley, Møller, and Waagepetersen (2000), second moment measure, Bartlett spectrum, cross-K function, cross-G function, J-function, and mark correlation function. Our aim is eventually to implement the vast majority of the statistical techniques described in the spatial statistics literature (e.g. Diggle (2003); Stoyan and Stoyan (1995)).

Parametric model-fitting: a key feature of **spatstat** is its generic algorithm for fitting point process models to data. The point process models to be fitted may be quite general Gibbs/Markov models; they may include inhomogeneous spatial trend, dependence on covariates, and interpoint interactions of any order (i.e. not restricted to pairwise interactions). Models are specified using a `formula` in the R language, and are fitted using a single function `ppm` analogous to `glm` and `gam`. A fitted model can be printed, plotted, predicted, updated, and simulated. Capabilities for residual analysis and model diagnostics will be added in version 1.6.

Simulation of point process models: **spatstat** can generate simulated realisations of a wide variety of stochastic point processes. Some process parameters (intensity function, cluster distribution) may be arbitrary, user-supplied functions in the R language. Markov point process models of a very general kind (including arbitrary spatial inhomogeneity and user-supplied interaction potential) are simulated using a fast `Fortran` implementation of the Metropolis-Hastings algorithm. Fitted model objects obtained from the model-fitting algorithm can be simulated directly by Metropolis-Hastings.

4. Demonstration

A few examples of **spatstat**'s capabilities are shown in the following transcript of an R session. A more extensive demonstration can be seen by installing the package and typing `demo(spatstat)`.

```
R> library(spatstat)
R> data(cells)
R> cells

     planar point pattern: 42 points
     window: rectangle = [0,1] x [0,1]

R> plot(cells)
R> plot(ksmooth.ppp(cells))
R> plot(Kest(cells))
```

These commands performed some exploratory analysis of the dataset `cells`. The last two lines displayed a kernel-smoothed estimate of the intensity, and an estimate of the K function.

```
R> fit <- ppm(cells, ~1, Strauss(r=0.1))
R> fit

   Stationary Strauss process
   beta
   290.4221
   interaction distance:   0.1
   Fitted interaction parameter gamma:
   [1] 0.0126

R> Xsim <- rmh(fit)
R> plot(Xsim)
```

This code fits a Strauss point process model to the `cells` data. The object `fit` is a fitted point process model. The code prints a summary of the fitted model, then simulates a realisation from this fitted model.

```
R> data(demopat)
R> plot(demopat, box=FALSE)
R> plot(split(demopat))
R> plot(alltypes(demopat, "K"))
```

This code analyzes the point pattern shown in Figure 2 which consists of points of two different types. The `split` command separates the dataset into two point patterns according to their types, which are then plotted separately. The `alltypes` command computes the bivariate ('cross') K function $K_{ij}(r)$ for each pair of types i, j and plots them as a 2×2 array of graphs.

```
R> pfit <- ppm(demopat, ~marks + polynom(x,y,2), Poisson())
R> plot(pfit)
```

The call to `ppm` fits a non-stationary Poisson point process to the data in Figure 2. The logarithm of the intensity function of the Poisson process is described by the R formula `~marks + polynom(x,y,2)` which represents a log-quadratic function of the cartesian coordinates, multiplied by a constant factor depending on the type of point. The last line plots the fitted intensity function as a perspective view of a surface.

5. Data types

The basic data types in **spatstat** are Point Patterns, Windows, and Pixel Images. A point pattern is a dataset recording the spatial locations of all 'events' or 'individuals' observed in a certain region. A window is a region in two-dimensional space. It usually represents the 'study area'. A pixel image is an array of "brightness" values for each grid point in a rectangular grid inside a certain region. It may contain covariate data (such as a satellite image) or it may be the result of calculations (such as kernel smoothing).

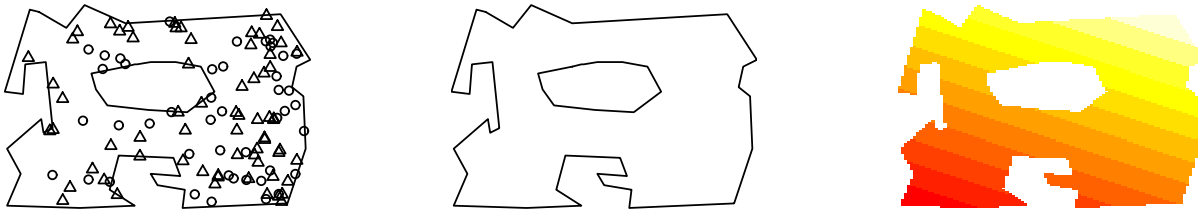

Figure 5: A point pattern, a window, and a pixel image.

spatstat uses the object-oriented features of R ("classes and methods") to make it easy to manipulate, analyse, and plot these datasets.

Note that there is no predetermined format for covariate data. Indeed that would be unnecessarily limiting, as there are many different kinds and formats of covariate information that might be needed. Our modelling and simulation code accepts covariate data in various formats.

5.1. Point patterns

A point pattern is represented in **spatstat** by an object of the class `"ppp"`. A dataset in this format contains the coordinates of the points, optional 'mark' values attached to the points, and a description of the spatial region or 'window' in which the pattern was observed.

To create a point pattern (class `"ppp"`) object we may create one from raw data using the function `ppp`, convert data from other formats (including other packages) using `as.ppp`, read data from a file using `scanpp`, manipulate existing point pattern objects using a variety of tools, or generate a random pattern using one of the simulation routines.

For example, to create a pattern of random points inside the rectangle $[0, 10] \times [0, 3]$,

```
R> x <- runif(20, max=10)
R> y <- runif(20, max=3)
R> u <- ppp(x, y, c(0,10), c(0,3))
```

The Venables and Ripley `spatial` library, which is part of the standard distribution of R, supplies a dataset `pines`. To convert this into our format,

```
R> library(spatial)
R> pines <- ppinit("pines.dat")
R> library(spatstat)
R> pines <- as.ppp(pines)
```

A point pattern must have a window

Note especially that, when you create a new point pattern object, you need to specify the spatial region or window in which the pattern was observed.

We believe that the observation window is an integral part of the point pattern. A point pattern dataset consists of knowledge about where points were *not* observed, as well as the

locations where they *were* observed. Even something as simple as estimating the intensity of the pattern depends on the window of observation. It would be wrong, or at least different, to analyze a point pattern dataset by "guessing" the appropriate window (e.g. by computing the convex hull of the points). An analogy may be drawn with the difference between sequential experiments and experiments in which the sample size is fixed *a priori*.

For situations where the window is really unknown, **spatstat** provides the function `ripras` to compute the Ripley-Rasson estimator of the window, given only the point locations Ripley and Rasson (1977).

Marked point patterns

Each point in a spatial point pattern may carry additional information called a 'mark'. For example, a pattern of points which are classified into two or more different types (on/off, case/control, species, colour, etc) may be regarded as a pattern of marked points, where the mark attached to each point indicates which type it is. Data recording the locations and heights of trees in a forest can be regarded as a marked point pattern where the mark attached to a tree's location is the height of the tree.

In our current implementation, the mark attached to each point must be a *single* value (which may be numeric, character, complex, logical, or factor). Many of the functions in **spatstat** for marked point patterns require that the mark attached to each point be either

- a **continuous variate** or "real number". An example is the Longleaf Pines dataset (`longleaf`) in which each tree is marked with its diameter at breast height. The `marks` component must be a `numeric` vector such that `marks[i]` is the mark value associated with the `i`th point. We say the point pattern has *continuous marks*.

- a **categorical variate**. An example is the Amacrine Cells dataset (`amacrine`) in which each cell is identified as either "on" or "off". Such point patterns may be regarded as consisting of points of different "types". The `marks` component must be a `factor` such that `marks[i]` is the label or type of the `i`th point. We call this a *multitype point pattern* and the levels of the factor are the possible types.

See Figures 6–7.

Note that, in some other packages, a point pattern dataset consisting of points of two different types (A and B say) is represented by two datasets, one representing the points of type A and another containing the points of type B. In **spatstat** we take a different approach, in which all the points are collected together in one point pattern, and the points are then labelled by the type to which they belong. An advantage of this approach is that it is easy to deal with multitype point patterns with more than 2 types. For example the classic Lansing Woods dataset represents the positions of trees of 6 different species. This is available in **spatstat** as a single dataset, a marked point pattern, with the marks having 6 levels.

Standard datasets

Some standard point pattern datasets are supplied with the package. They are summarised in Table 1.

Name	Description	Reference	Marks	Window	Covariates
`ants`	ant nests	Harkness and Isham (1983)	species (2)	polygon	subregions & line segments
`amacrine`	amacrine cells	Diggle (1986)	type (on/off)	rectangle	
`betacells`	retinal ganglia cells	Wässle, Boycott, and Illing (1981)	type (on/off)	rectangle	
`bramblecanes`	bramble canes	Hutchings (1979); Diggle (1983)	age (3 classes)	rectangle	
`cells`	biological cells	Ripley (1981)		rectangle	
`copper`	copper deposits	Berman (1986)		rectangle	line segments
`demopat`	artificial dataset		type (A/B)	polygon (with hole)	
`finpines`	pine trees		tree height	rectangle	
`hamster`	hamster tumour cells	Diggle (1983)	type (2)	rectangle	
`japanesepines`	pine trees	Numata (1964); Diggle (1983) Ogata and Tanemura (1984)		rectangle	
`lansing`	Lansing Woods data	Gerrard (1969) Cox (1979)	species (6)	rectangle	
`longleaf`	Longleaf Pines data	Platt *et al.* (1988) Rathbun and Cressie (1994)	tree diameter	rectangle	
`nztrees`	trees	Mark and Esler (1970); Ripley (1981)		rectangle	
`redwood`	redwood saplings	Strauss (1975); Ripley (1977)		rectangle	
`redwoodfull`	redwood saplings (full set)	Strauss (1975)		rectangle	subregions
`spruces`	spruce trees	Stoyan, Kendall, and Mecke (1987)	tree diameter	rectangle	
`swedishpines`	pine trees	Strand (1972); Ripley (1988)		rectangle	

Table 1: Point pattern datasets supplied in **spatstat** version 1.5-7.

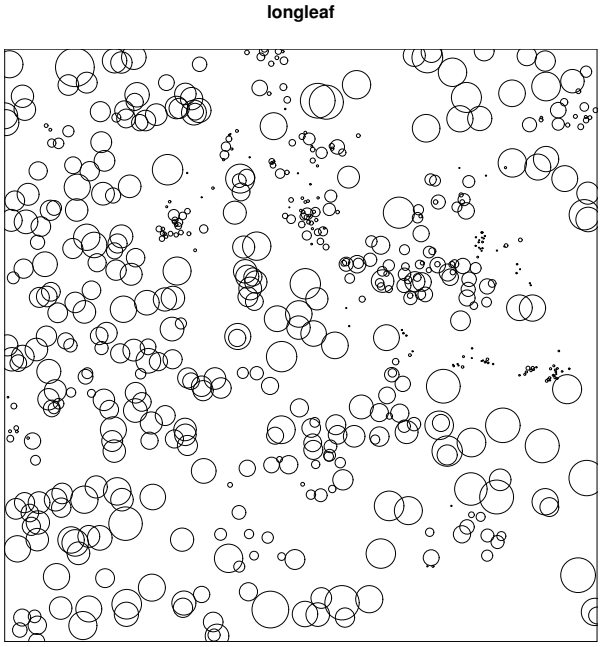

Figure 6: Point pattern with continuous marks (tree diameter). The Longleaf Pines dataset Platt *et al.* (1988); Rathbun and Cressie (1994), available as `longleaf`.

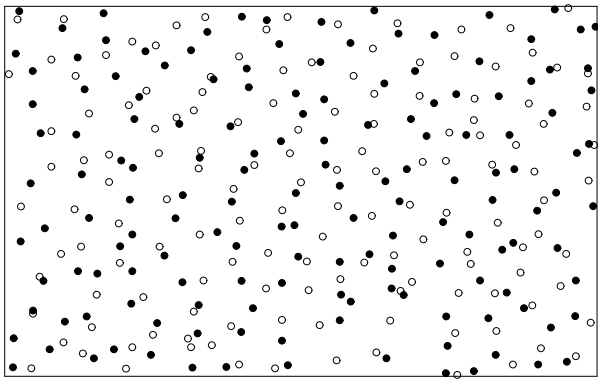

Figure 7: Point pattern with categorical marks (cell type). Hughes' amacrine cell dataset Diggle (1986), available as `amacrine`.

5.2. Windows

An object of the class `"owin"` (for "observation window") represents a spatial region or 'window' in the two-dimensional plane. A window usually represents our 'study area': the window of observation of a point pattern, or the region where we want to make predictions, etc.

To create a window object we can build one from data in R, using `owin` and other tools; extract the window from one of the point pattern datasets supplied with the package by typing `W <- X$window` where `X` is the point pattern; convert data from other formats using `as.owin`; manipulate existing windows using a wide variety of tools or derive a window from a point pattern or pixel image using various tools.

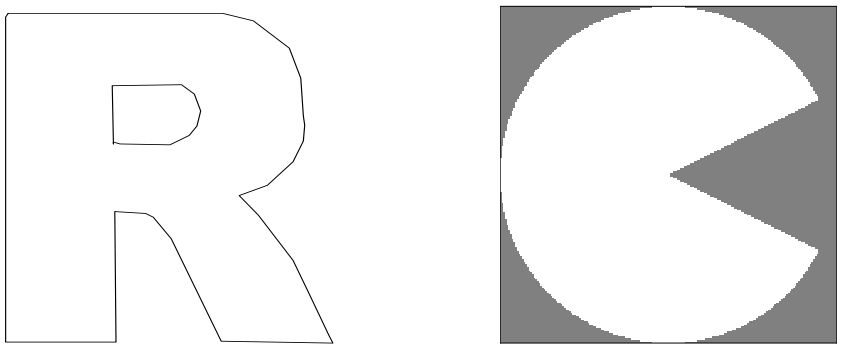

Figure 8: Polygonal window (left) and pixellated window (right).

The shape of a window is almost arbitrary; it may be a rectangle, a polygon, a collection of polygons (including holes), or a binary image mask. See Figure 8.

spatstat supports polygonal windows of arbitrary shape and topology. That is, the boundary of the window may consist of one or more closed polygonal curves, which do not intersect themselves or each other. The window may have 'holes'.

spatstat also supports 'pixellated' windows. A matrix with logical entries is interpreted as a binary pixel image whose entries are `TRUE` where the corresponding pixel belongs to the window. Pixellated windows can be created from raw data, read from data files, or created by analytic equations. They are also produced in **spatstat** by various geometrical operations, such as morphological erosion.

5.3. Pixel images

An object of the class `"im"` represents a pixel image. It is essentially a matrix of numerical values associated with a rectangular grid of points inside a window in the x, y plane. A pixel image may be displayed on the screen as a digital image, a contour map, or a relief surface. Image objects can be created explicitly using `im`. Data in other formats can be converted to an `"im"` object using `as.im`.

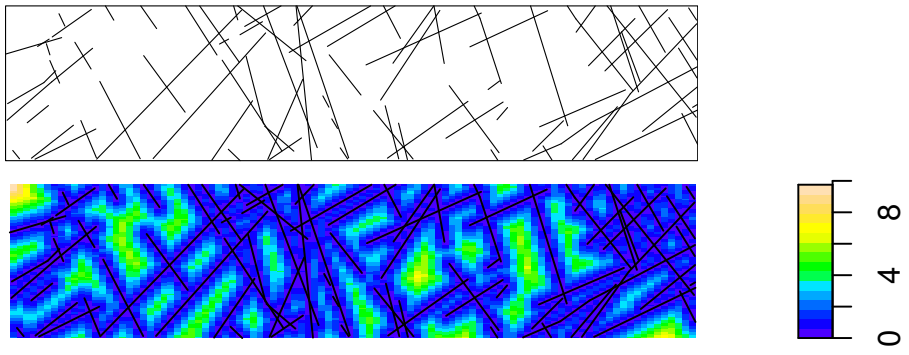

Figure 9: Example of pixel image data. *Top:* line segment pattern from the `copper` dataset. *Bottom:* a pixel image derived from the `copper` data. Pixel value is the distance to the nearest line segment.

A pixel image may contain real experimental data, for example, a satellite image of the study

region. One of the important roles of pixel images is to provide covariate data for statistical models. The brightness value of the image at a particular pixel is the value of the spatial covariate at that location. For example, Figure 9 shows a colour image derived from the spatial covariates in the `copper` dataset.

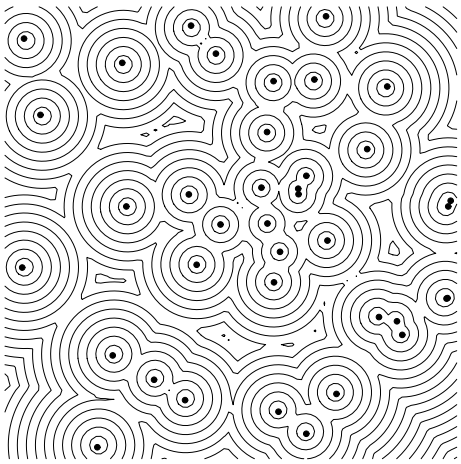

Figure 10: A computed pixel image (displayed as a contour plot): the distance transform of a point pattern. Obtained by `contour(distmap(X))` where `X` was the point pattern. Dots indicate original point pattern dataset.

Pixel images are also produced by many functions in **spatstat**, for example when we apply kernel smoothing to point pattern data (`ksmooth.ppp`), when we estimate the second moment measure of a point process (`Kmeasure`), compute the geometric covariance of a window (`setcov`) or evaluate the distance map of a point pattern (`distmap`). See Figure 10.

We also use pixel images to represent mathematical functions of the Cartesian coordinates. Any function object `f(x,y)` in R can be converted into a pixel image using `as.im`.

6. Operations on data

Once we have created a point pattern dataset, it can be inspected, plotted and modified using the commands described here.

6.1. Basic inspection of data

There are `print`, `summary` and `plot` methods for point patterns, windows, and pixel images.

```
R> hamster

   marked planar point pattern: 303 points
   multitype, with levels = dividing, pyknotic
   Window: rectangle = [ 0 , 1 ] x [ 0 , 1 ]

R> summary(hamster)
```

```
Marked planar point pattern: 303 points
Average intensity 303 points per unit area
Marks:
     frequency proportion intensity
dividing       226      0.746       226
pyknotic        77      0.254        77

Window: rectangle = [0,1] x [0,1]
Window area =  1
```

```
R> plot(hamster)
```

Plotting is isometric, i.e. the physical scales of the x and y axes are the same. For marked point patterns, the plotting behaviour depends on whether the marks are continuous or categorical, and typical displays are shown in Figures 6 and 7 respectively. To see the locations of the points without the marks, type `plot(unmark(X))`.

The colours, plotting characters, line widths and so on can be modified by adding arguments to the `plot` methods. Default plotting behaviour can also be controlled using the function `spatstat.options`.

The function `identify.ppp`, a method for `identify`, allows the user to examine a point pattern interactively.

6.2. Subsets of point patterns

spatstat supports the extraction of subsets of a point pattern, with a method for the indexing operator `"["`. This performs either "*thinning*" (retaining/deleting some points of a point pattern) or "*trimming*" (reducing the window of observation to a smaller subregion and retaining only those points which lie in the subregion).

If `X` is a point pattern object then `X[subset, ]` will cause the point pattern to be "thinned", retaining only the points indicated by `subset`. The latter can be any type of subset argument such as a positive integer vector, a logical vector, or a negative integer vector (the latter indicating which points should be deleted).

The pattern will be "trimmed" if we call `X[ , window]` where `window` is an object of class `"owin"`. Only those points of `X` lying inside the new `window` will be retained.

6.3. Other operations on point patterns

Marks can readily be added to and removed from a point pattern using the functions `unmark` and `setmarks` or the operator `%mark%`. Marks can be manipulated rapidly using the methods for `cut`, `split` and `split<-` for point patterns. For a point pattern with numerical marks, `cut.ppp` will transform the marks into factor levels. For a multitype point pattern, `split.ppp` will separate the dataset into a list of point patterns, each consisting of points of one type. The functions `superimpose` and `"split<-.ppp"` will combine several point patterns into a single point pattern, attaching mark labels if required.

Geometrical operations on point patterns include planar rotation, translation and affine transformation (`rotate, shift` and `affine`). There are functions to compute the distance

from each point to its nearest neighbour (`nndist`), the distance between each pair of points (`pairdist`) and the distance from each point to the boundary of the window (`bdist.points`).

6.4. Manipulating windows

The following functions are available for manipulating windows.

`bounding.box`	Find smallest rectangle enclosing the window with sides parallel to the x and y axes
`erode.owin`	Erode window by a distance r
`rotate.owin`	Rotate the window
`shift.owin`	Apply vector translation
`affine.owin`	Apply an affine transformation
`complement.owin`	Invert (inside $\leftrightarrow$ outside)
`is.subset.owin`	Test whether one window contains another
`trim.owin`	Intersect window with rectangle
`intersect.owin`	Intersection of windows
`union.owin`	Union of windows
`ripras`	Estimate window from points

Pixellating windows

The shape of any spatial region may be approximated by a binary pixel image. In **spatstat** the image is represented as a window object (class `"owin"`) of type `"mask"`. The following commands are useful.

`as.mask`	Convert to pixel approximation
`raster.x`	Extract the x coordinates of the pixel raster
`raster.y`	Extract the y coordinates of the pixel raster

The default accuracy of the approximation can be controlled using `spatstat.options`.

Additionally `nearest.raster.point` maps continuous cartesian coordinates to raster locations.

Geometrical computations with windows

The following commands are useful for computing geometrical quantities.

`inside.owin`	Test whether (x, y) points are inside window
`area.owin`	Compute window's area
`diameter`	Compute window's diameter
`eroded.areas`	Compute areas of eroded windows
`bdist.points`	Compute distances from data points to window boundary
`bdist.pixels`	Compute distances from all pixels to window boundary
`centroid.owin`	Compute centroid (centre of mass)
`distmap`	Compute distance transform of window

6.5. Pixel images

Functions which return a pixel image include the following.

`Kmeasure`	Reduced second moment measure of point pattern
`setcov`	Set covariance function of spatial window
`ksmooth.ppp`	Kernel smoothed intensity estimate of point pattern
`distmap`	Distance transform of point pattern

Functions which manipulate a pixel image include the following.

`im`	Create a pixel image
`as.im`	Convert data to pixel image
`plot.im`	Display as digital image
`contour.im`	Display as contour map
`persp.im`	Display as perspective view
`[.im`	Extract subset of pixel image
`shift.im`	Apply vector shift to pixel image
`print.im`	Print basic information
`summary.im`	Print summary
`is.im`	Test whether object is a pixel image

6.6. Programming tools

spatstat also contains some programming tools to assist in calculations with point patterns. One of these is the function `applynbd` which can be used to visit each point of the point pattern, identify its neighbouring points, and apply any desired operation to these neighbours.

For example the following code calculates the distance from each point in the pattern `redwood` to its second nearest neighbour:

```
R> nnd2 <- applynbd(redwood, N = 2, exclude=TRUE,
                    function(Y, cur, d, r){max(d)})
```

This has obvious applications for LISA methods Anselin (1995); Cressie and Collins (2001b,a). One can also use `applynbd` to perform animations in which each point of the point pattern is visited and a graphical display is executed. There is an example in `demo(spatstat)`.

7. Exploratory data analysis

The literature on spatial statistics contains a very large number of techniques for the exploratory analysis of point pattern data. Perhaps the most famous example is Ripley's K-function. As far as we know, the vast majority of these techniques have never been implemented in public domain software, apart from the initial 'proof-of-concept' implementations by their original authors. The uptake of new methods in practice seems to have been severely limited by the lack of such software. Accordingly, one of the main aims of the **spatstat** project is to implement the existing, published techniques of spatial statistics in open source software.

7.1. Initial inspection of data

Initial, interactive inspection of a point pattern dataset is supported by the methods for `print`, `summary`, `plot` and `identify` mentioned above. The function `summary.ppp` computes the average intensity of points, summarises the marks if `X` is a marked point pattern, and

describes the window. Subsets of the data can be extracted using the methods for `"["`, `cut` and `split`.

7.2. Spatial inhomogeneity

One of the important questions about a point pattern dataset is whether it can be treated as spatially homogeneous. To investigate this, Diggle and others have recommended kernel smoothing. The function `ksmooth.ppp` performs kernel smoothing of a point pattern, and yields a pixel image object.

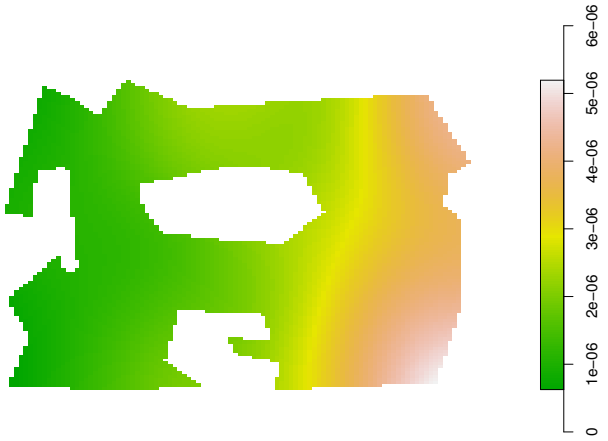

Figure 11: Kernel smoothed intensity estimate for the point pattern in Figure 2, indicating a clear trend from left to right.

spatstat contains several functions which extend classical techniques (developed for homogeneous patterns) to inhomogeneous point patterns. They include `Kinhom` (an inhomogeneous version of the K function Baddeley *et al.* (2000)) and the model-fitting function `ppm`.

7.3. Summary statistics for unmarked point patterns

Exploratory analysis of point patterns is based largely on summary statistics. The **spatstat** package will compute estimates of the summary functions

$F(r)$, the empty space function (contact distribution or 'point-to-event' distribution)

$G(r)$, the nearest neighbour distance distribution function ('event-to-event' distribution)

$J(r)$, the function $J = (1 - G)/(1 - F)$

$K(r)$, the reduced second moment function ("Ripley's K function")

$g(r)$, the pair correlation function $g(r) = [\frac{d}{dr}K(r)]/(2\pi r)$

for a point pattern, and their analogues for marked point patterns.

These estimates can be used for exploratory data analysis and in formal inference about a spatial point pattern. They are well described in the literature, e.g. Ripley (1981), Diggle

(2003), Cressie (1991), (Stoyan *et al.* 1995, Chapter 15), Stoyan and Stoyan (1995). The J-function was introduced in van Lieshout and Baddeley (1996).

The point pattern is assumed to be stationary (homogeneous under translations) in order that the functions F, G, J, K be well-defined and the corresponding estimators approximately unbiased. (There is an extension of the K function to inhomogeneous patterns; see below).

The corresponding **spatstat** library functions are:

`Fest`	estimate of empty space function F
`Gest`	estimate of nearest neighbour distribution function G
`Jest`	estimate of J-function
`Kest`	estimate of Ripley's K-function
`allstats`	estimates of all four functions F, G, J, K
`pcf`	estimate of pair correlation function g

(Some others are listed below).

In each of these commands, the user has a choice of several alternative estimation methods. These estimators are based on different 'edge corrections', or strategies for removing the bias due to 'edge effects', which arise because we only observe the point pattern inside a restricted spatial window. Several dozen alternative edge corrections have been published in the literature; see Baddeley (1998); Stoyan and Stoyan (1995) for surveys. Part of the **spatstat** project is to implement all of these proposed estimators so that they may be compared in practice.

The routines `Fest, Gest, Jest, Kest, pcf` each return an object of class `"fv"` (for "function value"). This is a data frame with some extra attributes indicating the recommended way of plotting the function, and other information. It is a convenient way of storing (particularly for use in future plotting) several different estimates of the same function.

A column labelled `r` in this data frame contains the values of the argument r for which the summary function ($\widehat{F}(r)$, etc) has been evaluated. Other columns give the estimates of the summary function itself, using several competing estimators. Along with the different function estimates, the data frame includes the vector of theoretical expected values (`theo`) that the function would have under the assumption of "complete spatial randomness" (CSR) i.e. under a homogeneous Poisson point process model.

There are methods for `print` and `plot` for the class `"fv"`. The `plot` method is particularly useful. It is a generalisation of `plot.formula`, and enables the summary functions to be re-plotted in a variety of ways.

There are various recommendations in the literature about how to plot the summary functions to reveal diagnostic information. An aim of **spatstat** is to make it easy to plot the summary functions in different ways.

Probably the most common exploratory graphic is a plot of $\widehat{K}(r)$ against r. An example of a useful transformed graphic is a plot of $L(r) = \sqrt{\widehat{K}(r)/\pi}$ against r, as recommended by Ripley (1981), the rationale being that this procedure linearizes the plot and stabilizes the variance. Diggle (1983, 2003) recommends plotting $\widehat{K}(r) - \pi r^2$ against r, so as to remove the mean. These plots can be achieved as follows:

```
R> Kc <- Kest(cells)
R> plot(Kc)
```

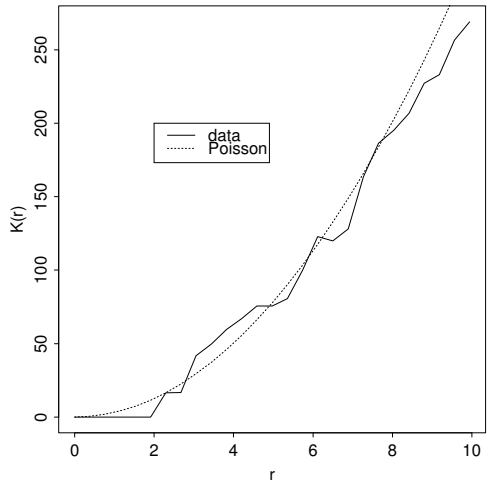

Figure 12: Output of `plot(Kest(X))`.

```
R> plot(Kc, cbind(r, sqrt(iso/pi)) ~ r)
R> plot(Kc, cbind(trans,iso,border) - theo ~ r)
```

Notice the use of `cbind` in the last two plots. The effect is that several functions (the columns in the `cbind` expression on the left hand side) will be plotted in the same plot, against the variable on the right hand side of the formula.

With respect to the empty space (contact) distribution function F, Ripley (1981, 1988) simply plots $\widehat{F}(r)$ against r, whereas Diggle (2003) plots $\widehat{F}(r)$ against $F_0(r) = 1 - \exp\{-\hat{\lambda}\pi r^2\}$, this being the form of F under the assumption of complete spatial randomness. This is in effect a P–P plot. Another useful graphic (suggested by Murray Aitkin) is a plot of $\sin^{-1}(\sqrt{\widehat{F}(r)})$ against $\sin^{-1}(\sqrt{F_0(r)})$. The function $g(x) = \sin^{-1}\sqrt{x}$ is Fisher's variance-stabilising transformation for the binomial estimator of a proportion, and indeed seems to approximately stabilise the variance in this context.

These alternative plots may be displayed as follows.

```
R> Fc <- Fest(cells)
R> plot(Fc)
R> plot(Fc, cbind(km, trans, border) ~ theo)
R> fisher <- function(x) { asin(sqrt(x)) }
R> plot(Fc, fisher(cbind(km, trans, border))
             ~ fisher(theo))
```

Initially it may be unclear which of the summary functions will provide insight, and it is usually desirable to calculate and plot estimates of all four. The command `plot(allstats(X))` will produce a plot of estimates of the four main summary functions K, F, G and J.

Distances between points are also computed (without edge correction) by:

`nndist`	nearest neighbour distances
`pairdist`	distances between all pairs of points
`exactdt`	distance from any location to nearest data point

There are also several related alternative functions. For the second order statistics, alternatives are:

`Kinhom`	K function for inhomogeneous point patterns
`Kest.fft`	fast K-function using FFT for large datasets
`Kmeasure`	reduced second moment measure

The function `Kmeasure` yields a pixel image of the estimated Reduced Second Moment Measure $\mathfrak{K}$ (see (Stoyan and Stoyan 1995, p. 245)). This measure is the Fourier transform of the Bartlett spectrum Bartlett (1964, 1975). Although first defined in the 1960's this concept appears not to have been implemented in software until recently. Its usefulness in data analysis is yet to be explored.

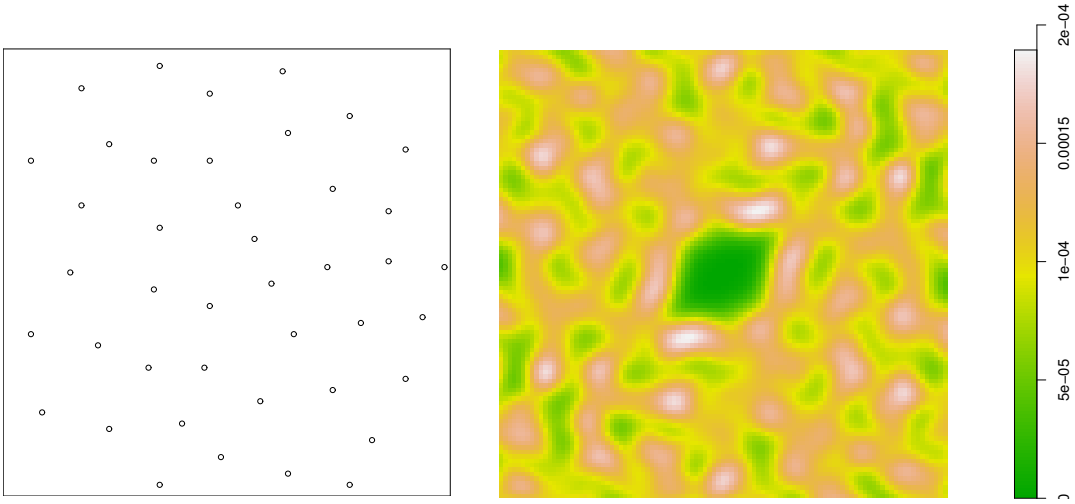

Figure 13: The `cells` dataset (Left) and a density estimate of its second moment measure (Right).

Figure 13 shows the well-known `cells` dataset, and a density estimate of its second moment measure, computed by `Kmeasure`. [The algorithm takes the raw Bartlett periodogram, multiplies by the Fourier transform of the bivariate normal density, then takes the inverse FFT to yield the smoothed density.] The large contour in the centre of the Figure is a region of low second moment density close to the origin, caused by the spatial inhibition between points at short distances. The pronounced non-circular shape of this contour suggests that the interpoint interaction is anisotropic, which does not appear to have been noticed before.

8. Summary statistics for multitype point patterns

Analogues of the G, J and K functions have been defined in the literature for "multitype" point patterns, that is, patterns in which each point is classified as belonging to one of a finite number of possible types (e.g. on/off, species, colour). The best known of these is the bivariate (cross) K function $K_{ij}(r)$ derived by counting, for each point of type i, the number

of type j points lying closer than r units away.

The corresponding nearest-neighbour function $G_{ij}(r)$ is the distribution of the distance from a typical point of type i to the nearest point of type j. Using the symbol $\bullet$ to denote points of any type (i.e. all points regardless of their type) we may define analogous functions $K_{i\bullet}$ and $G_{i\bullet}$. For further explanation see van Lieshout and Baddeley (1999).

`Gcross,Gdot,Gmulti`	multitype nearest neighbour distributions $G_{ij}, G_{i\bullet}$
`Kcross,Kdot, Kmulti`	multitype K-functions $K_{ij}, K_{i\bullet}$
`Jcross,Jdot,Jmulti`	multitype J-functions $J_{ij}, J_{i\bullet}$

These functions operate in a very similar way to `Gest, Jest, Kest` with additional arguments specifying the type(s) of points to be studied.

8.1. Function arrays

For multitype patterns we might want to compute a summary function for the points of type i for each of the possible types of the pattern. Alternatively we might want to compute a summary function for each possible pair of types (i, j).

A *function array* is a collection of functions $f_{i,j}(r)$ indexed by integers i and j. An example is the set of cross K functions $K_{ij}(r)$ for all possible pairs of types i and j in a multitype point pattern ($1 \leq i, j \leq m$ where m is the number of types). It is best to think of this as a genuine matrix or array.

A function array is represented in **spatstat** by an object of type `"fasp"` (function array for spatial patterns). It can be stored, plotted, indexed and subsetted in a natural way. If `Z` is a function array, then

```
R> plot(Z)
R> plot(Z[,3:5])
```

will plot the entire array, and then plot the subarray consisting only of columns 3 to 5.

The function `alltypes` will compute a summary statistic for each possible type, or each possible pair of types, in a multitype point pattern. The value returned by `alltypes` is a function array object.

For example if `X` is a multitype point pattern with 3 possible types,

```
R>  Z <- alltypes(X, "K")
```

yields a 3×3 function array such that (say) `Z[1,2]` represents the cross-type K function $K_{1,2}(r)$ between types 1 and 2.

The command `plot(Z)` will then plot the entire set of cross K functions as a two-dimensional array of plot panels. Arguments to `plot.fasp` can be used to change the plotting style, the range of the axes, and to select which estimator of K_{ij} is plotted. These options apply to all the plot panels simultaneously.

The command `allstats` yields a 2×2 function array containing the F, G, J and K functions of an (unmarked) point pattern.

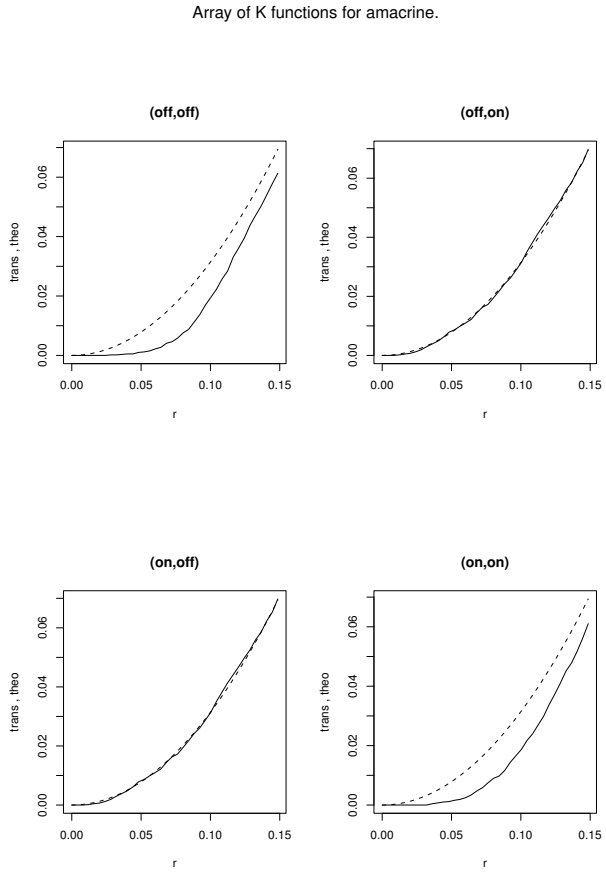

Figure 14: The result of `plot(alltypes(amacrine, "K"))`.

8.2. Summary functions for point patterns with continuous marks

Some point pattern datasets are marked, but not multitype. That is, the points may carry marks that do not belong to a finite list of possible types. The marks might be continuous numerical values, complex numbers, etc.

An example is the Longleaf Pines data (shown in Figure 6) where the marks represent tree diameters. In **spatstat**, a marked point pattern with numerical marks is plotted using circles of radius proportional to the positive marks, and squares of side length proportional to the negative marks.

There are a few ways to study such patterns in **spatstat**. The function `markcorr` computes the mark correlation function of an arbitrary marked point pattern. The functions `Kmulti, Gmulti, Jmulti` operate on arbitrary marked point patterns. They require arguments `I, J` identifying two subsets of the point pattern. These two subsets will be treated as two discrete types.

Alternatively a marked point pattern can be converted to a multitype point pattern using the function `cut.ppp`, for example, classifying the marks into High, Medium and Low. Then one can apply the abovementioned functions for multitype point patterns. This is usually a good exploratory step, along with the use of `split.ppp` to separate the sub-patterns. Of course one can also ignore the marks (using the function `unmark` to remove them) and analyse only

the locations of the points.

9. Model fitting

The most important feature of **spatstat** is its ability to fit parametric models of spatial point processes to point pattern data. The scope of possible models is very wide: they may include spatial trend, dependence on covariates, interpoint interactions of any order (i.e. we are not restricted to pairwise interactions), and dependence on marks.

Models are fitted by a function `ppm` which is analogous to `glm` and `lm`. The fitted model objects can be printed, plotted, predicted, and even simulated. Methods for computing residuals and plotting model diagnostics will be released in version 1.6.

Models are currently fitted by the method of maximum pseudolikelihood, using a computational device developed by Berman and Turner (1992) which we adapted to pseudolikelihoods in Baddeley and Turner (2000). Although maximum pseudolikelihood may be statistically inefficient, it has the virtue that we can implement it in software with great generality. Future versions of the package will implement other fitting methods.

9.1. Formulating models

The point process models fitted by `ppm` are formulated in terms of their *conditional intensity* rather than their likelihood. The (Papangelou) conditional intensity is a function $\lambda(u, \mathbf{x})$ of spatial location u and of the entire point pattern $\mathbf{x}$. See Baddeley and Turner (2000); Cox and Isham (1980) and the excellent surveys by Ripley (1988, 1989).

For example, the homogeneous Poisson process (complete spatial randomness, CSR) has conditional intensity

$$\lambda(u, \mathbf{x}) = \beta$$

where β is the expected number of points per unit area. The inhomogeneous Poisson process with intensity function $\beta(u)$ has conditional intensity

$$\lambda(u, \mathbf{x}) = \beta(u). \tag{1}$$

The Strauss process, a simple model of dependence between points, has conditional intensity

$$\lambda(u, \mathbf{x}) = \beta\gamma^{t(u,\mathbf{x})} \tag{2}$$

where $t(u, \mathbf{x})$ is the number of points of $\mathbf{x}$ that lie within a distance r of the location u. Here $r, \beta > 0$ and $\gamma \in [0, 1]$ are parameters.

Our technique fits any model which belongs to the regular exponential family of distributions and which has a conditional intensity. The conditional intensity can then be written in the form

$$\lambda(u, \mathbf{x}) = \exp(\theta^{\mathsf{T}} B(u) + \varphi^{\mathsf{T}} C(u, \mathbf{x})) \tag{3}$$

where θ, φ are the canonical parameters. Both θ and φ may be vectors of any dimension, corresponding to the dimensions of the vector-valued statistics $B(u)$ and $C(u, \mathbf{x})$ respectively.

The term $B(u)$ depends only on the spatial location u, so it represents "spatial trend" or spatial covariate effects. The term $C(u, \mathbf{x})$ represents "stochastic interactions" or dependence

between the points of the random point process. For example it is absent if the model is a Poisson process.

The Strauss process (2) with fixed interaction range r conforms to (3) if we set $\theta = \log \beta$, $\varphi = \log \gamma$, $B(u) \equiv 1$ and $c(u, \mathbf{x}) = t(u, \mathbf{x})$.

9.2. Fitting a model to data

Overview

The model-fitting function is called `ppm` and is strongly analogous to `lm` or `glm`. In simple usage, it is called in the form

```
ppm(X, trend, interaction, ...)
```

where `X` is the point pattern dataset, `trend` is an S language `formula` describing the spatial trend (the function $B(u)$ in equation 3), and `interaction` is an object of a special class `"interact"` describing the stochastic dependence between points in the pattern (the function $C(u, \mathbf{x})$ in equation (3)). Other arguments may provide covariates and control the fitting algorithm.

Thus, the function $B(u)$ in (3) is treated as the 'systematic' component of the model, and is described by a `formula` analogous to the formula for the linear predictor in a generalised linear model. In this analogy the link is always the logarithm, so the model formula in a `ppm` call is always a description of the **logarithm** of the conditional intensity.

The function $C(u, \mathbf{x})$ in (3) is regarded as a "distributional" component of the model analogous to the distribution family in a generalised linear model. It is described in **spatstat** by an object of class `"interact"` that we create using specialised **spatstat** functions, similar to those which create the `family` argument to `glm`.

For example

```
R> ppm(X, ~1, Strauss(r=0.1), ....)
```

fits the stationary Strauss process with interaction radius $r = 0.1$. The spatial trend formula `~1` is a constant, meaning the process is stationary. The argument `Strauss(r=0.1)` is an object representing the interpoint interaction structure of the Strauss process with interaction radius $r = 0.1$. Similarly

```
R>  ppm(X, ~x + y, Poisson())
```

fits the non-stationary Poisson process with a *loglinear* intensity of the form

$$\beta(x, y) = \exp(\theta_0 + \theta_1 x + \theta_2 y)$$

where $\theta_0, \theta_1, \theta_2$ are (scalar) parameters to be fitted, and x, y are the cartesian coordinates. Similarly a *log-quadratic* intensity in x,

$$\beta(x, y) = \exp(\theta_0 + \theta_1 x + \theta_2 x^2)$$

could be fitted by

```
R>  ppm(X, ~x + I(x^2), Poisson())
```

Spatial trend

The `trend` argument of `ppm` describes any spatial trend and covariate effects. The default is `~1`, which corresponds to a process without spatial trend or covariate effects. The formula `~x` corresponds to a spatial trend of the form $\lambda(x, y) = \exp(a + bx)$, while `~x + y` corresponds to $\lambda(x, y) = \exp(a + bx + cy)$ where x, y are the Cartesian coordinates. These could be replaced by any R language formula (with empty left hand side) in terms of the reserved names `x, y` and `marks`, or in terms of some spatial covariates which you must then supply to `ppm`. There is no restriction on the formula since the function $B(u)$ in (3) is arbitrary.

The trend formula may be an arbitrary expression involving the Cartesian coordinates. For example

```
R> ppm(X, ~ sqrt(x^2 + y^2), Poisson())
```

fits an inhomogeneous Poisson process with intensity decaying or increasing exponentially with distance from the origin, while

```
R> ppm(X, ~ factor(ifelse(x > 2, 0, 1)), Poisson())
```

fits an inhomogeneous Poisson process with different, constant intensities on each side of the line $x = 2$.

spatstat provides a function `polynom` which generates polynomials in 1 or 2 variables. For example

```
~ polynom(x, y, 2)
```

represents a polynomial of order 2 in the Cartesian coordinates x and y. This would give a "log-quadratic" spatial trend.[2] Similarly

```
~ harmonic(x, y, 2)
```

represents the most general *harmonic* polynomial of order 2 in x and y.

Other possibilities include B-splines and smoothing splines, fitted with `bs` and `s` respectively. These terms introduce smoothing penalties, and thus provide an implementation of "penalised maximum pseudolikelihood" estimation (cf. Divino, Frigessi, and Green (2000)). For example

```
R>  ppm(X, ~bs(x,2), Poisson())
```

fits a non-stationary Poisson process whose log conditional intensity is modelled by a B-spline with 2 degrees of freedom.

The special term `offset` can also be used in the trend formula. It has the same role in `ppm` as it does in other model-fitting functions, namely to add to the linear predictor a term which is not associated with a parameter. For example

[2] We caution against using the standard function `poly` for the same purpose here. For a model formula containing `poly`, prediction of the fitted model can be erroneous, for reasons which are well-known to R users. The function `polynom` provided in **spatstat** does not exhibit this problem.

```
~ offset(3 * sin(x))
```

will fit the model with log trend $\beta + 3 \sin x$ where β is the only parameter to be estimated.

It is slightly more tricky to include *observed* spatial covariates; see Section 9.6 below.

Interaction terms

The higher order ("interaction") structure can be specified using one of the following functions. They yield an object (of class `"interact"`) describing the interpoint interaction structure of the model.

`Poisson` ..... Poisson process
`Strauss` ..... Strauss process
`StraussHard` ..... Strauss process with a hard core
`Softcore` ..... Pairwise soft core interaction
`PairPiece` ..... Pairwise interaction, step function potential
`DiggleGratton` ... Diggle-Gratton potential
`LennardJones` .... Lennard-Jones potential
`Geyer` ..... Geyer's saturation process
`OrdThresh` ..... Ord's process, threshold on cell area

Note that `ppm` estimates only the "canonical" parameters of a point process model. These are parameters θ such that the loglikelihood is linear in θ, as in equation (3), possibly after a re-parametrisation.

Other so-called "irregular" parameters (such as the interaction radius r of the Strauss process) cannot be estimated directly by this technique, and their values must be specified a priori, as arguments to the interaction function. Profile pseudolikelihood Baddeley and Turner (2000) can be used to fit such parameters.

For more advanced use, the following functions will accept "user-defined potentials" in the form of an arbitrary S language function. They effectively allow arbitrary point process models of these three classes.

`Pairwise` .... Pairwise interaction, user-supplied potential
`Ord` ..... Ord model, user-supplied potential
`Saturated` ... Saturated pairwise model, user-supplied potential

9.3. Fitted models

The value returned by `ppm` is a "fitted point process model" of class `"ppm"`. It can be stored, inspected, plotted and predicted.

```
R> fit <- ppm(X, ~1, Strauss(r=0.1), ...)
R> fit
R> plot(fit)
R> pf <- predict(fit)
R> coef(fit)
```

Methods are provided for the following generic operations applied to `"ppm"` objects:

`print`	Print basic information
`summary`	Print extensive summary information
`coef`	Extract fitted model coefficients
`plot`	Plot fitted intensity
`fitted`	Compute fitted conditional intensity or trend at data points
`predict`	Compute predictions (spatial trend, conditional intensity)
`update`	Update the fit

Printing the fitted object `fit` will produce text output describing the fitted model in its traditional form. For example, the traditional parameters β, γ of the Strauss process (2) are not in canonical form (3). The print method back-transforms the fitted canonical parameter θ to the traditional parameters β and γ.

Plotting the fitted model object will display the spatial trend and the conditional intensity, as perspective plots, contour plots and image plots.

The `predict` method computes either the spatial trend or the conditional intensity. The default behaviour is to produce a pixel image of both trend and conditional intensity, where these are appropriate. The conditional intensity $\lambda(u, \mathbf{x})$ can be evaluated at any desired locations u, but $\mathbf{x}$ is taken to be the observed data pattern to which the model was fitted.

Examples of calls to `predict.ppm` are the following:

```
R> data(cells)
R> m <- ppm(cells,~polynom(x,y,2),Strauss(0.05),
                    rbord=0.05)
R> trend <- predict(m,type="trend",ngrid=100)
R> cif   <- predict(m,type="cif",ngrid=100)
```

The resulting objects `trend` and `cif` are pixel images. One could then plot the resulting surfaces with calls like

```
R> persp(trend,theta=-30,phi=40,d=4,
            ticktype="detailed",zlab="z")
R> persp(cif,theta=-30,phi=40,d=4,
            ticktype="detailed",zlab="z")
```

We note again that the result of `predict` may be incorrect if the point process model's trend component is expressed in terms of one of the functions `poly`, `bs`, `lo`, or `ns`.

The plot method (`plot.ppm`) will take a fitted point process model and plot the trend and/or the conditional intensity. By default this surface is calculated at a 40×40 grid of points on the (enclosing rectangle of) the observation window. The plots may be produced as perspective plots, images, or contour plots. A simple example is

```
R> plot(fit,cif=FALSE,how="persp")
```

Here `fit` is the model fitted to the Numata data on page 27.

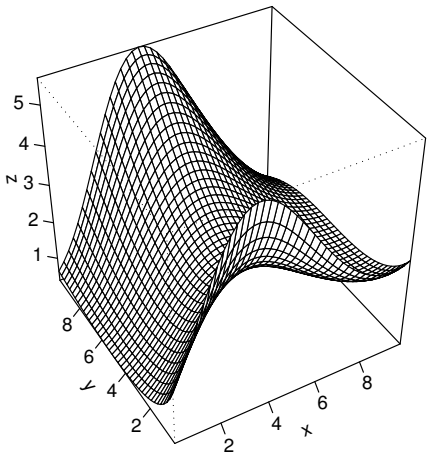

Figure 15: Fitted log-cubic trend for the full Numata pines data set obtained using `predict.ppm`.

9.4. In defence of maximum pseudolikelihood

Disadvantages of maximum pseudolikelihood (MPL) include its small-sample bias and inefficiency Besag (1977); Jensen and Møller (1991); Jensen and Künsch (1994) relative to maximum likelihood estimators (MLE).

However, as far as we are aware, there is currently no software implementation of any MLE technique for fitting point process models at the level of generality and flexibility that is achieved in **spatstat**. Numerical approximation methods Ogata and Tanemura (1986) and Markov Chain Monte Carlo methods Geyer and Møller (1994); Geyer (1999) are highly specific to the chosen model, and require careful tuning to ensure good performance. Markov Chain Monte Carlo is computationally intensive, especially for inhomogeneous spatial patterns, because of increased parameter dimensionality. Recent theoretical improvements Geyer and Møller (1994); Geyer (1999) have not yet led to better software implementations.

Thus, maximum pseudolikelihood has several advantages. It is extremely fast in execution, compared to the MLE. Our new implementation of the MPLE also enables new models to be specified very easily, and accommodates a wide range of models.

In real data analysis, the model should be regarded as tentative, and a fitted model should be criticised or validated, and possibly modified and re-fitted Chatfield (1988); Cox and Snell (1981); Davison and Snell (1991); Tukey (1977); Venables and Ripley (1997). The accuracy of the fitting procedure is not the only consideration. Hence, there is an important place for quick-and-dirty fitting methods especially in the analysis of real datasets.

When `ppm` is used, fitting a model encompassing both spatial inhomogeneity and interpoint interaction becomes routine. For example we can fit the model chosen for the Numata data in Ogata and Tanemura (1986) as follows:

```
R> fit <- ppm(numata,~polynom(x,y,3), Softcore(0.5))
```

In any case, more accurate estimation algorithms usually require a good starting value of the parameter estimate, and it is universally the MPL estimate which is used. Thus, an implementation of MPL is a prerequisite to implementing other techniques.

9.5. Fitting models to multitype point patterns

The function `ppm` will also fit models to multitype point patterns. A multitype point pattern is a point pattern in which the points are each classified into one of a finite number of possible types (e.g. species, colours, on/off states). In **spatstat** a multitype point pattern is represented by a `"ppp"` object `X` whose marks are a `factor`. Figure 7 shows an example.

Currently, `ppm` will not fit models to a marked point pattern if the marks are not a factor.

Trend component

The first-order component ("trend") of a multitype point process model may depend on the marks. For example, a stationary multitype Poisson point process could have different (constant) intensities for each possible mark. A general nonstationary process could have a different spatial trend surface for each possible mark.

In order to represent the dependence of the trend on the marks, the trend formula passed to `ppm` may involve the reserved name `marks`.

The trend formula `~1` states that the trend is constant and does not depend on the marks. The formula `~marks` indicates that there is a separate, constant intensity for each possible mark. The correct way to fit the multitype Poisson process is

```
R> ppm(X, ~ marks, Poisson())
```

The result of fitting this model to the data in Figure 7 yields the following output.

```
Stationary multitype Poisson process
Possible marks:
off on
Intensity:
Trend formula: ~marks

Fitted intensities:
beta_off  beta_on
88.68302 94.92830
```

This indicates that the fitted model is a multitype Poisson process with intensities 88.7 and 94.9 for the points of type "off" and "on" respectively.

Getting more elaborate, the trend formula might involve both the marks and the spatial locations or spatial covariates. For example the trend formula `~marks + polynom(x,y,2)` signifies that the first order trend is a log-quadratic function of the cartesian coordinates, multiplied by a constant factor depending on the mark.

The formulae

```
~ marks * polynom(x,2)
~ marks + marks:polynom(x,2)
```

both specify that, for each mark, the first order trend is a different log-quadratic function of the cartesian coordinates. The second form looks "wrong" since it includes a "`marks` by `polynom`" interaction without having `polynom` in the model, but since `polynom` is a covariate rather than a factor this is is allowed, and makes perfectly good sense. As a result the two foregoing models are in fact mathematically equivalent. However, the fitted model objects will give slightly different output.

For example, the first model `~marks * polynom(x,2)` fitted to the data in Figure 7 gives the following output (assuming `options("contrasts")` is set to its default, namely the 'treatment' contrasts):

```
Nonstationary multitype Poisson process
Trend formula: ~marks * polynom(x, 2)

Fitted coefficients for trend formula:
               (Intercept)                    markson
                 4.3127945                  0.2681231
         polynom(x, 2)[x]         polynom(x, 2)[x^2]
                 0.4651860                 -0.2363352
  markson:polynom(x, 2)[x] markson:polynom(x, 2)[x^2]
                -0.6781045                  0.4023491
```

This form of the model gives two quadratic functions: a "baseline" quadratic

$$P_0(x, y) = 4.3127945 + 0.4651860x - 0.2363352x^2$$

and a quadratic associated with the mark level "`on`",

$$P_{\mathrm{on}}(x, y) = 0.2681231 - 0.6781045x + 0.4023491x^2.$$

The baseline quadratic is the logarithm of the fitted trend for the points of type `off`, since `off` is the first level of the factor `marks`. For points of type `on`, since we are using the treatment contrasts, the log trend is

$$P_0(x, y) + P_{\mathrm{on}}(x, y) = 4.580918 - 0.2129185x + 0.1660139x^2.$$

On the other hand, when the second model `~marks + marks:polynom(x,2)` is fitted to the same dataset, the output is

```
Nonstationary multitype Poisson process
Trend formula: ~marks + marks:polynom(x, 2)

Fitted coefficients for trend formula:
                (Intercept)                     markson
                  4.3127945                   0.2681231
  marksoff:polynom(x, 2)[x]    markson:polynom(x, 2)[x]
                  0.4651860                  -0.2129185
marksoff:polynom(x, 2)[x^2]  markson:polynom(x, 2)[x^2]
                 -0.2363352                   0.1660138
```

This says explicitly that the log trend for points of type `off` is

$$Q_{\mathrm{off}}(x, y) = 4.3127945 + 0.4651860x - 0.2363352x^2$$

while for points of type `on` it is

$$Q_{\mathrm{on}}(x, y) = 4.580918 - 0.2129185x + 0.1660139x^2.$$

Hence the two fitted models are mathematically identical.

Interaction component

For the interaction component of a multitype point process model, any of the interaction structures listed above for unmarked point processes may be used. These interactions do not depend on the marks, only on the locations of the points. We have additionally defined two interactions which do depend on the marks:

`MultiStrauss`	multitype Strauss process
`MultiStraussHard`	multitype Strauss/hard core

For the multitype Strauss process, a matrix of "interaction radii" must be specified. If there are m distinct levels of the marks, we require a matrix `r` in which `r[i,j]` is the interaction radius r_{ij} between types i and j. For the multitype Strauss/hard core model, a matrix of "hardcore radii" must be supplied as well. These matrices will be of dimension $m \times m$ and must be symmetric.

9.6. Models with covariates

We can also fit point process models in which the point pattern is dependent on spatial covariates (e.g. altitude, soil pH, or distance to another spatial pattern). Any covariate data may be used, under the following conditions:

- the covariate must be a quantity $Z(u)$ observable (at least in principle) at each location u in the window. There may be several such covariates, and they may be continuous valued or factors.
- the values $Z(x_i)$ of Z at each point of the data point pattern must be available.
- the values $Z(u)$ at *some* other points u in the window must be available. [3]

Thus, it is not enough simply to observe the covariate values at the points of the data point pattern. For example, a dataset consisting of locations of trees in a forest and measurements of the soil acidity at these locations only, is not sufficient data to fit a model in which tree density depends on pH.

Covariate data are passed to the function `ppm` through the argument `covariates`. It may be either a data frame or a list of pixel images.

[3] The accuracy of the algorithm depends on the number of these points and on their spatial arrangement. For a good approximation to the pseudolikelihood, the density of these points should be high throughout the window.

(a) If `covariates` is a list of pixel images, then each image is assumed to contain the values of a spatial covariate. The names of the entries in the list should match the names of covariates used in the trend formula. For example:

```
R> ppm(X, ~ log(altitude) + pH,
        covariates=list(pH=phimage,altitude=image3))
```

(b) If `covariates` is a data frame, then the ith row of the data frame is expected to contain the covariate values for the ith 'quadrature point' (see below). The column names of the data frame should match the names of the covariates used in the trend formula. For example:

```
R> ppm(X, ~ log(altitude) + pH, covariates=cov.df)
```

where `cov.df` is a data frame with columns called `pH` and `altitude`.

Covariates in a list of images

The format (a), in which `covariates` is a list of images, would typically be used when the covariate values are already given on a fine grid (e.g. satellite image data, geological survey data) or are easy to obtain in this form.

Suppose `X` is a point pattern representing the locations of alpine ash trees, and `A` is a pixel image containing values of the terrain elevation in metres above sea level. Then

```
R> ppm(X, ~bs(alt, 2), covariates=list(alt=A))
```

fits the inhomogeneous Poisson process model where the density of trees depends on elevation through a B-spline with 2 degrees of freedom.

It is also convenient to supply covariates in the pixel image format when the covariate value can easily be computed at any location from existing data. For example, Figure 9 shows a pixel image of covariate values derived from the `copper` dataset. The covariate value $Z(u)$ at pixel u is the distance from u to the nearest line segment in the `copper` dataset.

Covariates in a data frame

Typically you would use the data frame format (b) if the values of the spatial covariates can only be observed at certain fixed locations. You need to force `ppm` to use these locations to fit the model.

This requires a little more information about the software. Our function `ppm` is an implementation of the algorithm of Baddeley and Turner (2000) which is based on a quadrature technique originated by Berman and Turner (1992). Very briefly, a 'quadrature scheme' in **spatstat** comprises both *'data points'* (the points of the observed point pattern) and *'dummy points'* (some other locations in the window). It is usually created using the function `quadscheme`.

You will need to create a quadrature scheme based on the spatial locations where the covariate Z has been observed. Then the values of the covariate at these locations are passed to `ppm` through the data frame `covariates`.

For example, suppose that `X` is the observed point pattern and we are trying to model the effect of soil acidity (pH). Suppose we have measured the values of soil pH at the points x_i

of the point pattern, and stored them in a vector `XpH`. Suppose we have measured soil pH at some other locations u in the window, and stored the results in a data frame `U` with columns `x, y, pH`. Then do as follows:

```
R> Q <- quadscheme(data=X, dummy=list(x=U$x, y=U$y))
R> df <- data.frame(pH=c(XpH, U$pH))
```

Then the rows of the data frame `df` correspond to the quadrature points in the quadrature scheme `Q`. To fit just the effect of pH, type

```
R> ppm(Q, ~ pH, Poisson(), covariates=df)
```

where the term `pH` in the formula `~ pH` agrees with the column label `pH` in the argument `covariates = df`. This will fit an inhomogeneous Poisson process with intensity that is a loglinear function of soil pH. You can also try (say)

```
R> ppm(Q, ~ pH, Strauss(r=1), covariates=df)
R> ppm(Q, ~ factor(pH > 7), Poisson(), covariates=df)
R> ppm(Q, ~ polynom(x, 2) * factor(pH > 7), covariates=df)
```

9.7. Offset terms

As we mentioned in section 9.2, the special term `offset` can also be used in the trend formula. It has the same role in `ppm` as it does in other model-fitting functions, namely to add to the linear predictor a term which is not associated with a parameter.

This is specially useful in case-control studies. For example, suppose we have a spatial epidemiological dataset containing a point pattern `X` of the locations of cases of a rare disease, and another point pattern `Y` of 'controls' which are a sample from the susceptible population. We want to model `X` as a point process with intensity proportional to the local density ρ of the susceptible population. We estimate ρ by taking a kernel-smoothed estimate of the intensity of `Y`. Thus

```
R> rho.hat <- ksmooth.ppp(Y, sigma=1.2)
R> ppm(X, ~offset(log(rho)), covariates=list(rho=rho.hat))
```

The first line computes the values of the kernel-smoothed intensity estimate at a fine grid of pixels, and stores them in the pixel image object `rho.hat`. The second line fits the Poisson process model with log intensity

$$\log \lambda(u) = \theta + \log \rho(u)$$

where θ is an unknown parameter; that is, it fits the Poisson model with intensity

$$\lambda(u) = \beta \, \rho(u)$$

where $\beta = e^{\theta}$ is the only parameter to be estimated. Note that `covariates` must be a list of images, even though there is only one covariate. The variable name `rho` in the model formula must match the name `rho` in the list.

10. Simulation

Stochastic simulation of point process models is another area where the richness of the theoretical literature contrasts with the scarcity of stable public domain software.

The **spatstat** package now has substantial functionality for simulating point processes. In particular it has the powerful ability to simulate a pattern directly from a fitted model.

The following functions generate random point patterns from various stochastic models. We follow the R naming convention that a random generator is called `r`*dist* where *dist* is the name of the probability distribution. These functions return a point pattern (as an object of class `"ppp"`).

`runifpoint`	n independent random points uniform in window
`rpoint`	n independent random points with given density
`rmpoint`	n independent multitype random points with given density
`rpoispp`	(in)homogeneous Poisson process
`rmpoispp`	multitype (in)homogeneous Poisson
`rMaternI`	Matérn Model I inhibition process Matérn (1986)
`rMaternII`	Matérn Model II inhibition process Matérn (1986)
`rSSI`	Simple Sequential Inhibition
`rNeymanScott`	general Neyman-Scott process
`rMatClust`	Matérn Cluster process Matérn (1986)
`rThomas`	Thomas process
`rmh`	run Metropolis-Hastings algorithm

See Diggle (2003); Møller and Waagepetersen (2003); Stoyan and Stoyan (1995) for information about these models. For example

```
R> plot(rMaternI(200,0.05))
```

will plot one realisation of the Matérn Model I inhibition process with parameters $\beta = 200$ and $r = 0.05$. See Figure 16.

The function `rmh` enables simulation from a broad range of point process models. It will also simulate any of the *fitted* models obtained from `ppm`, with a few exceptions (currently `LennardJones` and `OrdThresh`.)

10.1. Poisson processes

Simulation of Poisson processes is effected by the function `rpoispp`. This will simulate both homogeneous and inhomogeneous Poisson processes, in arbitrary windows — polygons, polygons with "holes", and masks, as well as in rectangles. The intensity function of the Poisson process may be specified by a constant, by an R language function of the coordinates x and y, or by a pixel image. For example, the commands

```
R> lambda <- function(x, y) { 100 * exp( 3 * x) }
R> X <- rpoispp(lambda, win=square(1))
```

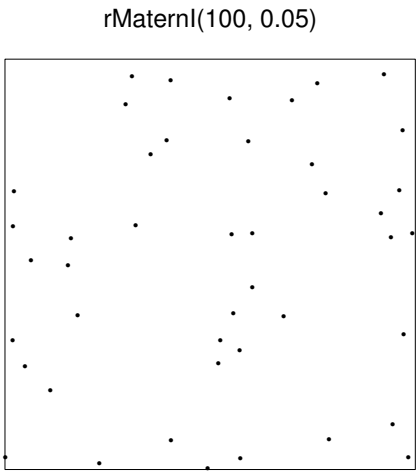

Figure 16: Realisation of Matérn Model I process.

generate a realisation of the Poisson process with intensity function $\lambda(x, y) = 100e^{3x}$ in the unit square.

There is also a function `rpoint` which generates a *fixed* number of points in a given window, again of arbitrary shape, with an arbitrary probability density. The probability density may again be specified by a constant, a `function(x,y)`, or a pixel image, and does not need to be normalised.

Examples:

```
R> data(letterR)
R> X1 <- rpoispp(100,win=letterR)
R> X2 <- rpoint(100,win=letterR)
```

The point pattern `X1` will have about 370 points in it since the area of `letterR` is about 3.7 units. (To find this out one can use `area.owin(letterR)`.) The point pattern `X2` will have exactly 100 points.

10.2. Doubly-stochastic processes

In a "doubly stochastic" or two-stage model, we first generate points at random according to a simple mechanism (usually the Poisson process) and then modify the pattern by randomly deleting or adding or moving some of the points.

The doubly stochastic mechanisms in **spatstat** for generating point patterns are the Matérn I and II models, the Matérn cluster process, the Thomas process, and the general Neyman-Scott process with bounded cluster radius. The corresponding function names are `rMaternI`, `rMaternII`, `rMatClust`, `rThomas` and `rNeymanScott`, respectively. The first two mechanisms are inhibitory — in each case a Poisson process is generated and then thinned. The other three mechanisms generate cluster processes, where the centres of the clusters form a Poisson process. The third and fourth mechanisms are special cases of the fifth.

Examples of usage of these functions:

```
R> X1 <- rMaternI(20, 0.05)
R> X2 <- rMaternII(20, 0.05)
R> X3 <- rMatClust(10, 0.05, 4)
R> X4 <- rThomas(10, 0.2, 5)
R> nclust <-  function(x0,y0,radius,n){ runifdisc(n, radius, x0, y0)}
R> X5 <- rNeymanScott(10, 0.2, nclust, radius=0.2, n=5)
```

Note the last example. The Neyman-Scott algorithm accepts an arbitrary random point process mechanism for generating the clusters.

10.3. Sequential processes

In 'Simple Sequential Inhibition', points are added to the window one-by-one, subject to the constraint that each new point must lie at least r units away from any existing point. The procedure terminates when it is not possible to add any more points satisfying this constraint. Termination must occur after a (random) finite number of steps. This process is implemented in **spatstat** as `rSSI`.

```
R> X <- rSSI(0.05, 200)
```

10.4. Metropolis-Hastings algorithm

The function `rmh` is an implementation of the Metropolis-Hastings algorithm for simulating point processes. A wide range of different processes can be simulated.

The function `rmh` is generic and has two methods, `rmh.default` which simulates a point process model specified explicitly by a list of its parameters, and `rmh.ppm` which simulates a point process model that has been fitted to data by the fitting function `ppm`. This latter ability is very useful in the context of Monte Carlo inference techniques.

```
R> m <- list(cif="strauss",
          par=c(beta=2,gamma=0.2,r=0.7),
          w=c(0,10,0,10))
R> X1 <- rmh(model=m,start=list(n.start=80), control=list(nrep=5e6))
R> fit <- ppm(cells, ~x, Strauss(0.1))
R> X2 <- rmh(fit,start=list(n.start=200), control=list(nrep=1e5))
```

In theory, any point process model which can be specified by means of a computable conditional intensity function can be simulated by Metropolis-Hastings, subject to some conditions to ensure convergence of the Markov Chain. But, for efficiency, the underlying calculations should be performed in dynamically loaded, compiled code (we use `Fortran 77`). So one would naively imagine that, each time we want to simulate a new model, we would have to write a new subroutine to calculate the model's conditional intensity function, and compile and link this to the existing code. This would appear to make it impossible to simulate an arbitrary fitted point process model automatically, since the spatial trend component (the `trend` argument to `ppm`) is specified by an arbitrary R language formula.

Fortunately there is a solution. The Metropolis-Hastings algorithm requires a series of random 'proposal' points. We have implemented the Metropolis-Hastings algorithm so that, instead

of generating its proposal points internally, it accepts as input a sequence of proposal points which are expected to be uniformly distributed in the simulation window W. Now suppose we want to simulate a point process X with conditional intensity

$$\lambda_X(u, \mathbf{x}) = \exp(\theta^\mathsf{T} B(u) + \varphi^\mathsf{T} C(u, \mathbf{x})).$$

Consider the related, simpler, point process Y with conditional intensity

$$\lambda(u, \mathbf{x}) = \exp(D + \varphi^\mathsf{T} C(u, \mathbf{x})) \tag{4}$$

where D is a constant. If we run the Metropolis-Hastings algorithm for the simpler process Y, but we supply it a sequence of proposal points generated from the non-uniform distribution with probability density

$$f(u) \propto \exp(\theta^\mathsf{T} B(u)),$$

then it can be shown that this is equivalent to running the Metropolis-Hastings algorithm for the desired process X. In other words, it is sufficient to implement the Metropolis-Hastings algorithm for **stationary** point processes with conditional intensities of the form (4).

The implementation of `rmh` in version 1.5 of **spatstat** can currently generate simulated realisations of the Strauss process; Strauss process with a hard core; the Soft Core process; Spatial's saturation process; pairwise interaction processes proposed by Diggle, Gratton and Stibbard (in `rmh.default` only) and by Diggle and Gratton; and multitype versions of the Strauss and Strauss/hard core processes. It can also generate processes with an arbitrary pairwise interaction function given as a vector of values or as a `stepfun` object. All these processes may have a spatial trend, which can be specified either explicitly as a `function(x,y)` or a pixel image, or implicitly as the trend in a fitted point process model.

11. Model checking

The reader may have noticed that **spatstat** does not provide some common statistical functionality such as

- confidence intervals or standard errors for summary statistics
- model diagnostics
- goodness-of-fit tests.

In fact this is a failing of the literature rather than the software. There are no simple expressions for confidence intervals and no simple methods for testing goodness-of-fit for a *fitted* spatial point process model.

What are commonly referred to as "confidence bands" for the K function are in fact the *critical bands* for a Monte Carlo test of a fixed null hypothesis Ripley (1981). Usually the null hypothesis assumes the point pattern is a realisation of a uniform Poisson process.

Simulation-based techniques Ripley (1981); Geyer (1999); Møller and Waagepetersen (2003) can be used to provide estimates of standard errors and confidence bands, and to perform goodness-of-fit tests of a fitted model. However, their implementation depends sensitively on the type of model, and their application is computationally intensive.

A general method of residual analysis and model diagnostics for fitted point process models was recently developed by Baddeley, Turner, Møller, and Hazelton (2004). This will be added to **spatstat** soon.

12. Future development

The **spatstat** package currently contains about 170 user-level functions, making it one of the largest contributed packages available for R. It is growing at the rate of about 30 new user-level functions per year.

The next major addition will be a capability for residual analysis and diagnostics for fitted point process models, using the methods of Baddeley *et al.* (2004).

The `ppm` function has an argument `method` which selects the parameter estimation technique. Currently the only option is `method="mpl"` representing Maximum Pseudolikelihood. The next step will be to implement the Huang and Ogata (1999) approximation to Maximum Likelihood.

Other planned additions include perfect simulation of some point process models Møller and Waagepetersen (2003), anisotropic summary statistics Stoyan and Stoyan (1995), LISA methods Anselin (1995), and Monte Carlo tests.

User feedback is a vital part of the development process. We welcome feedback and suggestions from all users. We also appreciate receiving new datasets for inclusion in the package.

Acknowledgements

spatstat was written by Adrian Baddeley and Rolf Turner, using substantial contributions of code from Marie-Colette van Lieshout, and with contributions from J.B. Chen, Y.C. Chin, S. Eglen, P. Grabarnik, U. Hahn, M.B. Hansen, K. Hornik, J. Mateu, L.S. Nielsen, B.D. Ripley, B. Rowlingson, A. Sarkka, K. Schladitz, B.T. Scott, M. Stevenson, B. Turlach, and A. van Burgel.

References

Anselin L (1995). "Local Indicators of Spatial Association – LISA." *Geographical Analysis*, **27**, 93–115.

Baddeley A (1998). "Spatial sampling and censoring." In O Barndorff-Nielsen, W Kendall, M van Lieshout (eds.), "Stochastic Geometry: Likelihood and Computation," chapter 2, pp. 37–78. Chapman and Hall, London.

Baddeley A, Møller J, Waagepetersen R (2000). "Non- and Semiparametric Estimation of Interaction in Inhomogeneous Point Patterns." *Statistica Neerlandica*, **54**(3), 329–350.

Baddeley A, Turner R (2000). "Practical Maximum Pseudolikelihood for Spatial Point Patterns (with discussion)." *Australian and New Zealand Journal of Statistics*, **42**(3), 283–322.

Baddeley A, Turner R (2005a). "Implementing Algorithms for Spatial Point Pattern Analysis in R." In preparation.

Baddeley A, Turner R (2005b). "Modelling Spatial Point Patterns in R." Submitted for publication.

Baddeley A, Turner R, Møller J, Hazelton M (2004). "Residual Analysis for Spatial Point Processes." *Research Report 2004/08*, School of Mathematics and Statistics, University of Western Australia. Submitted for publication.

Bartlett M (1964). "The Spectral Analysis of Two-Dimensional Point Processes." *Biometrika*, **51**, 299–311.

Bartlett M (1975). *The Statistical Analysis of Spatial Pattern*. Chapman and Hall, London.

Berman M (1986). "Testing for Spatial Association Between a Point Process and Another Stochastic Process." *Applied Statistics*, **35**, 54–62.

Berman M, Turner T (1992). "Approximating Point Process Likelihoods with GLIM." *Applied Statistics*, **41**, 31–38.

Besag J (1975). "Statistical Analysis of Non-lattice Data." *The Statistician*, **24**, 179–195.

Besag J (1977). "Efficiency of Pseudolikelihood Estimation for Simple Gaussian Fields." *Biometrika*, **64**(3), 616–618.

Bivand R (2001). "More on Spatial Data." *R News*, **1**(3), 13–17. URL http://CRAN.R-project.org/doc/Rnews/.

Chatfield C (1988). *Problem Solving: A Statistician's Guide*. Chapman and Hall.

Cliff A, Ord J (1981). *Spatial Processes: Models and Applications*. Pion, London.

Cox D, Snell E (1981). *Applied Statistics: Principles and Examples*. Chapman and Hall.

Cox DR, Isham V (1980). *Point Processes*. Chapman and Hall, London.

Cox T (1979). "A Method for Mapping the Dense and Sparse Regions of a Forest Stand." *Applied Statistics*, pp. 14–19.

Cressie N (1991). *Statistics for Spatial Data*. John Wiley and Sons, New York.

Cressie N, Collins L (2001a). "Analysis of Spatial Point Patterns using Bundles of Product Density LISA Functions." *Journal of Agricultural, Biological and Environmental Statistics*, **6**, 118–135.

Cressie N, Collins L (2001b). "Patterns in Spatial Point Locations: Local Indicators of Spatial Association in a Minefield with Clutter." *Naval Research Logistics*, **48**, 333–347.

Davison A, Snell E (1991). "Residuals and Diagnostics." In D Hinkley, N Reid, E Snell (eds.), "Statistical Theory and Modelling (in honour of Sir David Cox FRS)," chapter 4, pp. 83–106. Chapman and Hall, London.

Diggle P (1983). *Statistical Analysis of Spatial Point Patterns*. Academic Press, London.

Diggle P (1986). "Displaced Amacrine Cells in the Retina of a Rabbit: Analysis of a Bivariate Spatial Point Pattern." *Journal of Neuroscience Methods*, **18**, 115–125.

Diggle P (2003). *Statistical Analysis of Spatial Point Patterns*. Arnold, second edition.

Divino F, Frigessi A, Green P (2000). "Penalised Pseudolikelihood Estimation in Markov Random Field Models." *Scandinavian Journal of Statistics*, **27**(3), 445–458.

Gerrard D (1969). "Competition Quotient: A New Measure of the Competition Affecting Individual Forest Trees." *Research Bulletin 20*, Agricultural Experiment Station, Michigan State University.

Geyer C (1999). "Likelihood Inference for Spatial Point Processes." In O Barndorff-Nielsen, W Kendall, M van Lieshout (eds.), "Stochastic Geometry: Likelihood and Computation," Number 80 in Monographs on Statistics and Applied Probability, chapter 3, pp. 79–140. Chapman and Hall / CRC, Boca Raton, Florida.

Geyer C, Møller J (1994). "Simulation Procedures and Likelihood Inference for Spatial Point Processes." *Scandinavian Journal of Statistics*, **21**(4), 359–373. ISSN 0303-6898.

Harkness R, Isham V (1983). "A Bivariate Spatial Point Pattern of Ants' Nests." *Applied Statistics*, **32**, 293–303.

Harte D (2003). "Statistical Seismology Library (**SSLib**)." URL http://homepages.paradise.net.nz/david.harte/SSLib/

Högmander H, Särkkä A (1999). "Multitype Spatial Point Patterns with Hierarchical Interactions." *Biometrics*, **55**, 1051–1058.

Huang F, Ogata Y (1999). "Improvements of the Maximum Pseudo-Likelihood Estimators in Various Spatial Statistical Models." *Journal of Computational and Graphical Statistics*, **8**(3), 510–530.

Hutchings M (1979). "Standing Crop and Pattern in Pure Stands of *Mercurialis Perennis* and *Rubus Fruticosus* in Mixed Deciduous Woodland." *Oikos*, **31**, 351–357.

Isham V (1984). "Multitype Markov Point Processes: Some Approximations." *Proceedings of the Royal Society of London, Series A*, **391**, 39–53.

Jensen J, Künsch H (1994). "On Asymptotic Normality of Pseudo-Likelihood Estimates for Pairwise Interaction Processes." *Annals of the Institute of Statistical Mathematics*, **46**, 475–486.

Jensen J, Møller J (1991). "Pseudolikelihood for Exponential Family Models of Spatial Point Processes." *Annals of Applied Probability*, **1**, 445–461.

Mark A, Esler A (1970). "An Assessment of the Point-centred Quarter Method of Plotless Sampling in some New Zealand Forests." *Proceedings of the New Zealand Ecological Society*, **17**, 106–110.

Matérn B (1986). *Spatial Variation*. Number 36 in Lecture Notes in Statistics. Springer Verlag, New York.

Møller J, Waagepetersen R (2003). *Statistical Inference and Simulation for Spatial Point Processes*. Chapman and Hall/CRC, Boca Raton.

Moore M (ed.) (2001). *Spatial Statistics: Methodological Aspects and Applications*. Number 159 in Lecture Notes in Statistics. Springer.

Numata M (1964). "Forest Vegetation, Particularly Pine Stems in the Vicinity of Choshi — Coastal Flora and Vegetation in Choshi, Chiba Prefecture, IV (in Japanese)." *Bulletin of the Choshi Marine Laboratory*, (6), 27–37. Chiba University.

Ogata Y, Tanemura M (1981). "Estimation of Interaction Potentials of Spatial Point Patterns through the Maximum Likelihood Procedure." *Annals of the Institute of Statistical Mathematics*, **B 33**, 315–338.

Ogata Y, Tanemura M (1984). "Likelihood Analysis of Spatial Point Patterns." *Journal of the Royal Statistical Society, Series B*, **46**, 496–518.

Ogata Y, Tanemura M (1986). "Likelihood Estimation of Interaction Potentials and External fields of inhomogeneous Spatial Point Patterns." In I Francis, B Manly, F Lam (eds.), "Pacific Statistical Congress," pp. 150–154. Elsevier.

Peng RD (2003). "Multi-dimensional Point Process Models in R." *Journal of Statistical Software*, **8**(16), 1–27.

Platt WJ, Evans GW, Rathbun SL (1988). "The Population Dynamics of a Long-Lived Conifer (Pinus Palustris)." *The American Naturalist*, **131**, 491–525.

Rathbun SL, Cressie N (1994). "A Space-Time Survival Point Process for a Longleaf Pine Forest in Southern Georgia." *Journal of the American Statistical Association*, **89**, 1164–1173.

Ripley B (1976). "The Second-order Analysis of Stationary Point Processes." *Journal of Applied Probability*, **13**, 255–266.

Ripley B (1977). "Modelling Spatial Patterns (with discussion)." *Journal of the Royal Statistical Society, Series B*, **39**, 172–212.

Ripley B (1981). *Spatial Statistics*. John Wiley and Sons, New York.

Ripley B (1988). *Statistical Inference for Spatial Processes*. Cambridge University Press.

Ripley B (1989). "Gibbsian Interaction Models." In D Griffiths (ed.), "Spatial Statistics: Past, Present and Future," pp. 1–19. Image, New York.

Ripley BD (2001). "Spatial Statistics in R." *R News*, **1**(2), 14–15. URL http://CRAN.R-project.org/doc/Rnews/.

Ripley BD, Rasson J-P (1977). "Finding the Edge of a Poisson Forest." *Journal of Applied Probability*, **14**, 483–491.

Rowlingson B, Diggle P (1993). "Splancs: Spatial Point Pattern Analysis Code in S-PLUS." *Computers and Geosciences*, **19**, 627–655.

Särkkä A (1993). *Pseudo-Likelihood Approach for Pair Potential Estimation of Gibbs Processes*. Number 22 in Jyväskylä Studies in Computer Science, Economics and Statistics. University of Jyväskylä.

Stoyan D, Kendall W, Mecke J (1987). *Stochastic Geometry and its Applications*. John Wiley and Sons, Chichester.

Stoyan D, Kendall W, Mecke J (1995). *Stochastic Geometry and its Applications*. John Wiley and Sons, Chichester, second edition.

Stoyan D, Stoyan H (1995). *Fractals, Random Shapes and Point Fields*. Wiley.

Strand L (1972). "A Model for Stand Growth." In "IUFRO Third Conference Advisory Group of Forest Statisticians," pp. 207–216. INRA, Institut National de la Recherche Agronomique, Paris.

Strauss D (1975). "A Model for Clustering." *Biometrika*, **63**, 467–475.

Takacs R, Fiksel T (1986). "Interaction Pair-Potentials for a System of Ants' Nests." *Biometrical Journal*, **28**, 1007–1013.

Tukey J (1977). *Exploratory Data Analysis*. Addison-Wesley, Reading, Mass.

Upton G, Fingleton B (1985). *Spatial Data Analysis by Example. Volume I: Point Pattern and Quantitative Data*. John Wiley and Sons, Chichester.

van Lieshout M (2000). *Markov Point Processes and their Applications*. Imperial College Press.

van Lieshout M, Baddeley A (1996). "A Nonparametric Measure of Spatial Interaction in Point Patterns." *Statistica Neerlandica*, **50**, 344–361.

van Lieshout M, Baddeley A (1999). "Indices of Dependence Between Types in Multivariate Point Patterns." *Scandinavian Journal of Statistics*, **26**, 511–532.

Venables W, Ripley B (1997). *Modern Applied Statistics with S-PLUS*. Springer, second edition.

Wässle H, Boycott B, Illing RB (1981). "Morphology and Mosaic of On- and Off-Beta Cells in the Cat Retina and some Functional Considerations." *Proceedings of the Royal Society London Ser. B*, **212**, 177–195.

Affiliation:

Adrian Baddeley
School of Mathematics & Statistics
University of Western Australia
Nedlands WA 6009
Australia
E-mail: adrian@maths.uwa.edu.au
URL: http://www.maths.uwa.edu.au/~adrian/

Rolf Turner
Department of Mathematics and Statistics
University of New Brunswick
Fredericton, New Brunswick
Canada E3B 5A3
E-mail: rolf@math.unb.ca
URL: http://erdos.math.unb.ca/~rolf/

Journal of Statistical Software
January 2005, Volume 12, Issue 6.
http://www.jstatsoft.org/

Submitted: 2004-11-10
Accepted: 2005-01-26