3 Point patterns

Data \((s_1, s_2, \ldots, s_n)\) complete list of locations in \(\mathcal{R}\) at which events of interest have occurred (e.g., all trees in \(\mathcal{R}\) satisfying inclusion criterion).

Term ‘event’ originates from literature on time domain point processes; useful because often need to distinguish events from other arbitrary locations (‘points’) in \(\mathcal{R}\).

Simple point pattern has only locations; marked point pattern also some attributes attached to the events. Examples are

- type of event (tree species, nature of crime); multitype pattern
- continuous mark (size of object, duration of event)
- time of occurrence; spatio-temporal pattern
An example of a marked point pattern

Trees in a 50m × 50m sample plot in Lapland, diameters of symbols proportional to dbh (ranging in 26mm–441mm; smaller ones not measured). Thanks to Helena Henttonen!!

An example of a marked point pattern

Trees in a 50m × 50m sample plot in Lapland, diameters of symbols proportional to dbh (ranging in 26mm–441mm; smaller ones not measured). Thanks to Helena Henttonen!!
Tree map was produced by

```r
> trees1 <- read.table('trees1.dat',
+   col.names=c('x','y','z','species','dbh'))
> plot(trees1$x,trees1$y,type='n',axes=F,xlab='',ylab='',
+   asp=1,xlim=c(-40,25))
> polygon(c(-25,25,25,-25),c(-25,-25,25,25),lwd=3)
> species <- c(1,2,4,9)
> for(i in 1:4)
+   symbols(trees1$x[trees1$species==species[i]],
+     trees1$y[trees1$species==species[i]],
+     circles=trees1$dbh[trees1$species==species[i]]/200,
+     inches=F,fg=i,lwd=2,add=T)
> legend(-40,25,c('mänty','kuusi','koivu','raita'),pch='o',col=1:4)
> dev.off()
```

from file trees1.dat of following format

```
13.759 24.817 -0.724 4 135
13.626 23.23 -0.714 2 344
...```

56
Another tree map from Lapland

All pines, $dbh \in [72mm,367mm]$
Objectives of analysis

Basic interest: Do the events exhibit any systematic pattern, as opposed to being distributed at random in $\mathcal{R}$? General possibilities:

- clustering
- regularity

Examples of more specific questions

- Over what spatial scale?
- Are clusters associated with proximity to event(s) of other type?
- Is the pattern in one type of event similar to that of another?
- Are events which are clustered in space also close in time?
Difference to other types of spatial data

Patterns in the *locations* of the events are of interest. In other words, stochastic process we are studying relates to *where* events are likely to occur.

In later chapters (on spatially continuous phenomena and area data) analysis concerned with patterns in *attribute* i.e., stochastic process under study describes the spatial behaviour of values of the attribute.
Some ways of observing point patterns

Complete list of event locations is sometimes referred to as *mapped point pattern*. Other possibilities:

- *sampled point pattern*
- *aggregated point pattern*, e.g., quadrat counts
- *distance measurements*

Sometimes these incomplete ways of measuring point patterns are referred to as *field methods*. 
**Spatial point process**

Statistical model whose realisations are spatial point patterns (particular case of general ‘spatial stochastic process’). Simple (unmarked) point process can be represented as

$$\{Y(\mathcal{A}), \mathcal{A} \subseteq \mathcal{R}\},$$

where random variable $Y(\mathcal{A})$ counts the number of events occurring in $\mathcal{A}$. This representation not very useful as such, since $E(Y(\mathcal{A}))$ and $\text{Cov}(Y(\mathcal{A}), Y(\mathcal{A}'))$ depend on the size of the particular areas involved; instead characterise process by limiting behaviour of random variables $Y(\mathcal{A})$ ‘per unit area’ as $\mathcal{A}$ diminishes into a point.
3.1 First order properties

First order properties of spatial point process described in terms of *intensity* function

\[ \lambda(s) = \lim_{|ds| \to 0} \left\{ \frac{E(Y(ds))}{|ds|} \right\}, \]

where \( ds \) small region around \( s \)

- \( E(Y(\mathcal{A})) = \int_{\mathcal{A}} \lambda(s) \, ds \)

- for stationary process \( \lambda(s) = \lambda \) constant over \( \mathcal{R} \) and \( E(Y(\mathcal{A})) = \lambda |\mathcal{A}| \)
**Relationship to probability density estimation**

Intensity is closely related to *probability density*

\[
\text{intensity} = \text{density} \times \text{number of points per unit area}
\]

Hence methods developed for (bivariate) density estimation (or, more generally, exploration of probability distributions) often applicable

**but** need to be careful with *edge corrections*

- probability densities are often assumed to have *unbounded* support and
- it is usually assumed that no *censoring* occurs, i.e., there are no pre-specified bounds for the values of observations (or these are far enough from possible values to be inconsequential)
- therefore no edge corrections are usually needed
- in typical point pattern analysis censoring by region boundaries \(\implies\) edge corrections *are* needed
Quadrat counts

Partition $\mathcal{R}$ into sub-regions of equal area and use the counts of the number of events in each of these quadrats to summarise the pattern (corresponds to histograms)

Example for pines from the mixed species plot in Lapland

Note! This works only for square $\mathcal{R}$; can be easily modified for rectangular $\mathcal{R}$, but not so easily for general polygonal $\mathcal{R}$.

```r
> pines1 <- trees1[trees1$species==1,]
> breaks <- seq(-25,25,length=5)
> library(sm)  # non-standard library
> counts <- binning(pines1[,c('x','y')],
+  breaks=matrix(rep(breaks,2),ncol=2))$table.freq
> counts[4:1,]
[1,]  3   1   0   1
[2,]  0   1   1   3
[3,]  0  13   3   1
[4,]  2   4   6   2
```
we have effectively transformed our point pattern into area data & could use their methods in further analysis

however, we have also thrown away much of the spatial detail

smaller quadrats to retain more information \(\Rightarrow\) high variability in quadrat counts & meaningful interpretation difficult
Counts in moving windows

Window of suitable size is moved over a fine grid of locations in $\mathcal{R}$, and intensity at each grid point is estimated from the event count per unit area within the window centered on that point.

**Problem:** no account is taken of the relative locations of events within window.
Here’s an implementation I have modified from kde2d of the MASS library; note there is no edge correction.

> movcount
function (x, y, h, n = 25, lims = c(range(x), range(y)))
{
    nx <- length(x)
    if (length(y) != nx)
        stop("Data vectors must be the same length")
    gx <- seq(lims[1], lims[2], length = n)
    gy <- seq(lims[3], lims[4], length = n)
    h <- h/2
    ax <- outer(gx, x, "-")/h[1]
    ay <- outer(gy, y, "-")/h[2]
    z <- matrix(abs(ax)<1, n, nx) %*% t(matrix(abs(ay)<1, n, nx))
    return(list(x = gx, y = gy, z = z))
}
> mapines <- movcount(pines1$x, pines1$y, h=rep(12.5, 2), n=100, + lims=rep(c(-25,25), 2))
> summary(as.numeric(mapines$z))

   Min. 1st Qu. Median  Mean 3rd Qu.   Max. 
    0.00 0.000   1.000 2.284 3.000 15.000

> postscript('movcount.eps', width=5, height=5)
> library(splancs)
> plot(as.points(pines1),asp=1, type='n', + axes=F, xlab='', ylab='')
> image(mapines,col=gray((22:7)/22),add=T)
> polygon(c(-25,25,25,-25), c(-25,-25,25,25), + lwd=3)
> pointmap(as.points(pines1), + add=T,pch=16,col=2)
> polygon(+ pines1$x[18]+c(-12.5,12.5,12.5,-12.5)/2,+ pines1$y[18]+c(-12.5,-12.5,12.5,12.5)/2)
> dev.off()
Kernel estimation

Natural extension of moving counts taking account the relative locations of events within window by *weighted* counts: Points closer to window centre are weighted more.

This is clearly the most common way to explore first order effects.
Formally,

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

where

- \( s_1, \ldots, s_n \) are the locations of the observed events
- \( k \) is a bivariate probability density function (symmetric about the origin) called *kernel* in this context
- \( \tau > 0 \) is a smoothing parameter known as the *bandwidth*
- \( \delta_\tau(s) \) is an edge correction—the volume under the scaled kernel centred on \( s \) which lies inside \( R \).

Choice of \( k \) doesn’t matter much in practice, but \( \tau \) is crucial; rules of thumb, such as \( \tau = 0.68n^{-0.2} \) when \( R \) is unit square, have been suggested; however, one should *always* experiment with different values of \( \tau \).
Examples of kernel estimates
using splancs. Left: larger bandwidth, right: smaller bandwidth
These were produced by

```r
> library(splancs)
> postscript('kernel.eps', width=8, height=5)
> par(mfrow=c(1,2), mar=rep(.5,4))
> plot(as.points(pines1), asp=1, type='n', axes=F, xlab='', ylab='')
> image(kernel2d(cbind(pines1$x, pines1$y),
+ cbind(c(-25, 25, 25, -25), c(-25, -25, 25, 25)),
+ 10, 100, 100), col=gray((22:7)/22), add=T)
> polygon(c(-25, 25, 25, -25), c(-25, -25, 25, 25), lwd=3)
> pointmap(as.points(pines1), add=T, pch=16, col=2)
> plot(as.points(pines1), asp=1, type='n', axes=F, xlab='', ylab='')
> contour(kernel2d(cbind(pines1$x, pines1$y),
+ cbind(c(-25, 25, 25, -25), c(-25, -25, 25, 25)),
+ 8, 100, 100), add=T)
> polygon(c(-25, 25, 25, -25), c(-25, -25, 25, 25), lwd=3)
> pointmap(as.points(pines1), add=T, pch=16, col=2)
> dev.off()
```
Kernel estimates

with spatstat. Left: larger bandwidth, right: smaller bandwidth
These were produced by

```r
> postscript('kernel2.eps', width=8, height=5)
> par(mfrow=c(1,2), mar=rep(.5,4))
> plot(as.points(pines1), asp=1,
> + type='n', axes=F, xlab=' ', ylab=' ')
> image(ksmooth.ppp(as.ppp(cbind(pines1$x, pines1$y),
> + c(-25, 25, -25, 25)), 6), col=gray((22:7)/22), add=T)
> polygon(c(-25, 25, 25, -25), c(-25, -25, 25, 25), lwd=3)
> pointmap(as.points(pines1), add=T, pch=16, col=2)
> plot(as.points(pines1), asp=1, type='n', axes=F, xlab=' ', ylab=' ')
> contour(ksmooth.ppp(as.ppp(cbind(pines1$x, pines1$y),
> + c(-25, 25, -25, 25)), 4), add=T)
> polygon(c(-25, 25, 25, -25), c(-25, -25, 25, 25), lwd=3)
> pointmap(as.points(pines1), add=T, pch=16, col=2)
> dev.off()
```
Kernel estimates

with \texttt{kde2d} of MASS library. This is meant for \textit{density estimation} and does not include edge correction (see the difference?). Also the values on contour lines are densities rather than intensities.
These were produced by

```r
> library(MASS)
> par(mfrow=c(1,2),mar=rep(.5,4))
> pines1 <- trees1[trees1$species==1,]
> plot(pines1$x,pines1$y,
+   pch=16,col=2,xlab='',ylab='',axes=F,asp=1)
> polygon(c(-25,25,25,-25),c(-25,-25,25,25),lwd=3)
> contour(
+   kde2d(pines1$x,pines1$y),
+   add=T)
> plot(pines1$x,pines1$y,
+   pch=16,col=2,xlab='',ylab='',axes=F,asp=1)
> polygon(c(-25,25,25,-25),c(-25,-25,25,25),lwd=3)
> contour(
+   kde2d(pines1$x,pines1$y,h=rep(15,2)),
+   add=T)
```