Spatial point processes: Theory and practice illustrated with R

Ege Rubak

Department of Mathematical Sciences
Aalborg University

Lecture I, February 17, 2011

About the lecture series

- First, I would like to credit Adrian Baddeley for much of the material presented in this lecture series, which is heavily inspired by the workshop "Analysing spatial point patterns in R" he developed. Also, he and Rolf Turner developed the R-package `spatstat` which is the software we are using.
- The theory will be introduced from the very beginning from a practical point of view without many mathematical details.
- To illustrate the usage of R and `spatstat` some commands will be printed on the slides. For example to install and load `spatstat` run:

```
> install.packages(spatstat)
> library(spatstat)
```

- Please ask any questions you like during the lectures (however, I cannot guarantee that you will get an answer...).

Contents of lecture I

- Introduction to spatial point pattern data and some real data examples which contain both covariates and marks.
- Basic handling of data in `spatstat`.
- Introduction to point processes.
- First order properties for point processes.
- Simple non-parametric methods for point patterns.
- The homogeneous point process.

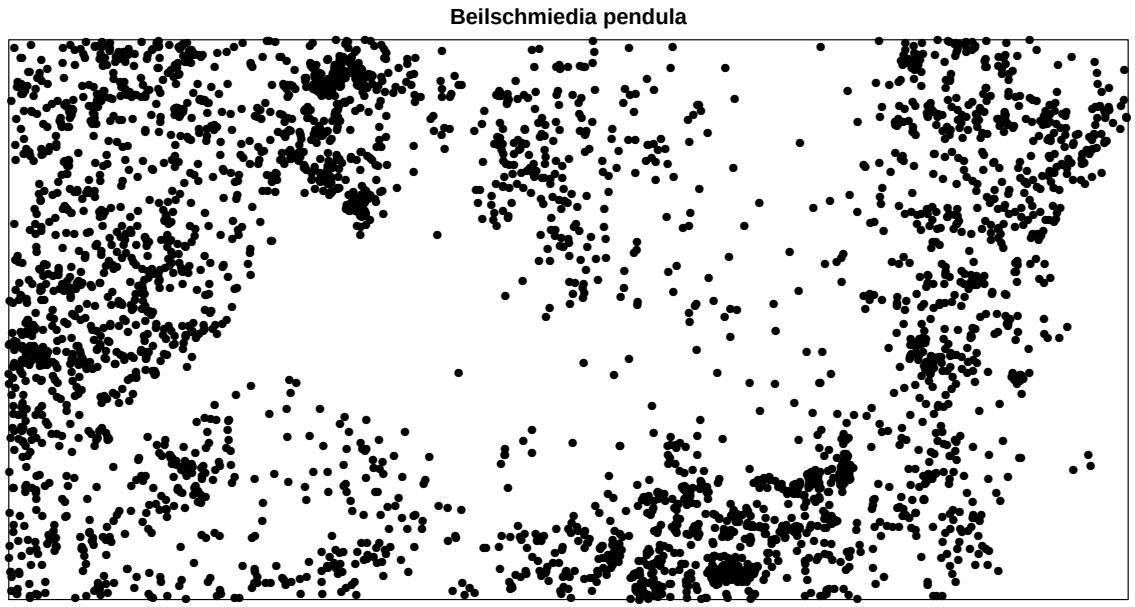
Spatial point pattern data

- A point pattern dataset gives the locations $\mathbf{x} = \{x_1, \ldots, x_n\}$ of objects/events occurring in a study region W (which we also refer to as an observation window).
- Points can represent trees, animal nests, earthquake epicentres, domiciles of new cases of a disease, galaxies, ...
- In principle the points can be situated in quite general spaces, but some of the most common examples are in $\mathbb{R}$, $\mathbb{R}^2$, $\mathbb{R}^3$, S^2 (e.g. the Earth's surface) or $\mathbb{R}^3 \times \mathbb{R}_+$ (e.g. space-time coordinates of earthquake epicentre location and time).
- We will work with points in $\mathbb{R}^2$. Among other things this is convenient because a lot of data are in 2D, the 2D data are easy to visualize, and the `spatstat` package was originally developed for this case (but 3D support is growing).

Tropical rainforest trees

The `spatstat` package includes some datasets which we can easily load and plot. The dataset `bei` consists of the locations of 3605 trees of the species "Beilschmiedia pendula" in a $1000\,\mathrm{m} \times 500\,\mathrm{m}$ study region of a tropical rainforest.

```
> data(bei)
> plot(bei, main = "Beilschmiedia pendula", pch = 16)
```



Point patterns in spatstat

In spatstat planar point patterns have the class ppp, and they can be plotted (as we have seen), printed etc.:

```
> class(bei)
```

```
[1] "ppp"
```

```
> bei
```

```
 planar point pattern: 3604 points
window: rectangle = [0, 1000] x [0, 500] metres
```

Among other things we can easily extract coordinates:

```
> xy <- coords(bei)
> xy[1:3, ]
```

```
      x     y
1  11.7 151.1
2 998.9 430.5
3 980.1 433.5
```

Observation windows in spatstat

The observation window is of class owin and can be obtained from a ppp-object:

```
> as.owin(bei)

window: rectangle = [0, 1000] x [0, 500] metres
```

Rectangular windows are easy to create:

```
> W <- owin(xrange = c(0, 500), yrange = c(0, 250))
> W

window: rectangle = [0, 500] x [0, 250] units
```

Windows can be used to extract sub-patterns:

```
> bei[W]

 planar point pattern: 709 points
window: rectangle = [0, 500] x [0, 250] units
```

Covariate information for point patterns

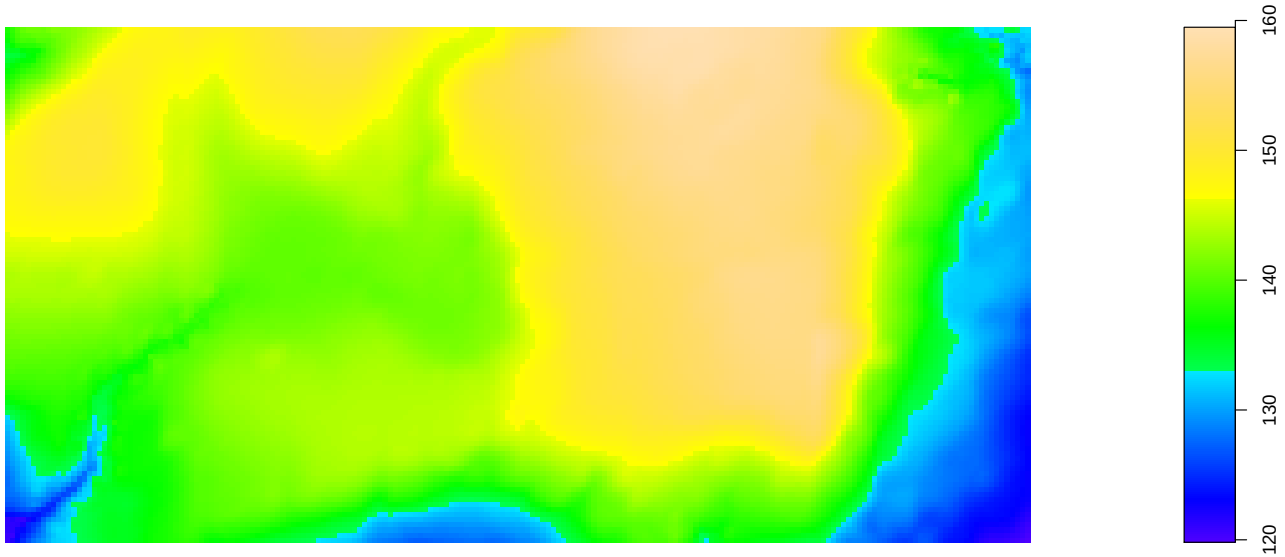
A point pattern dataset may also include covariate information. Two common types of covariate information are:

- A spatial function $Z(u)$ defined at all spatial locations $u \in W$, e.g. terrain elevation. Notice that we need to know Z in the entire observation window and not only at the data points.
- A spatial pattern such as another point pattern, or a line segment pattern, e.g. the location of some fixed object of interest in the observation window. Typically the covariate pattern would be used to define a surrogate spatial function Z, for example, $Z(u)$ may be the distance from u to the fixed covariate location.

Covariate for rainforest data (Elevation)

For the rainforest data we have information about the elevation of the terrain in form of a pixel image of class `im` in `spatstat`:

```
> elev <- bei.extra$elev
> class(elev)
[1] "im"
> plot(elev, main = "")
```



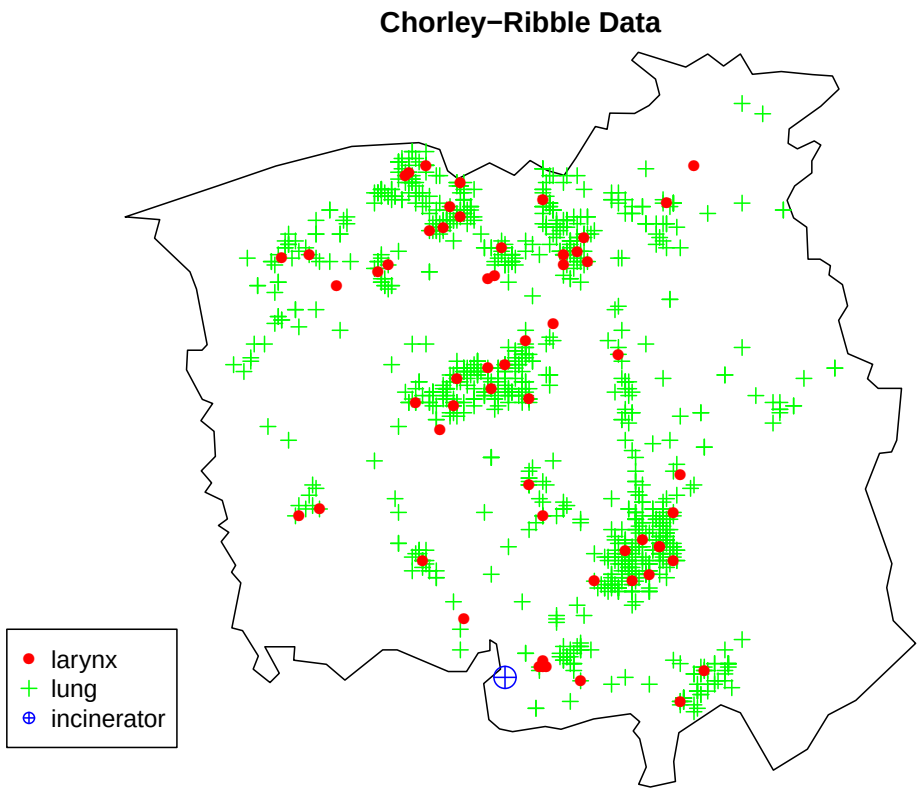
Marked point patterns

The points may have extra information called marks attached to them. The mark represents an "attribute" of the point. Typical mark variables are:

- Categorical marks, e.g. tree species or disease type.
- Continuous marks, e.g. tree diameter.
- Multivariate, e.g. both tree diameter and tree species.

Chorley-Ribble cancer cases

A marked point pattern of the domicile locations of cancer patients. The marks identify the cancer type as larynx or lung. We also have covariate information in form of the location of an industrial incinerator.



Marked point patterns in spatstat

Marked point patterns are also of class ppp. Basic mark information is revealed when the ppp-object is printed:

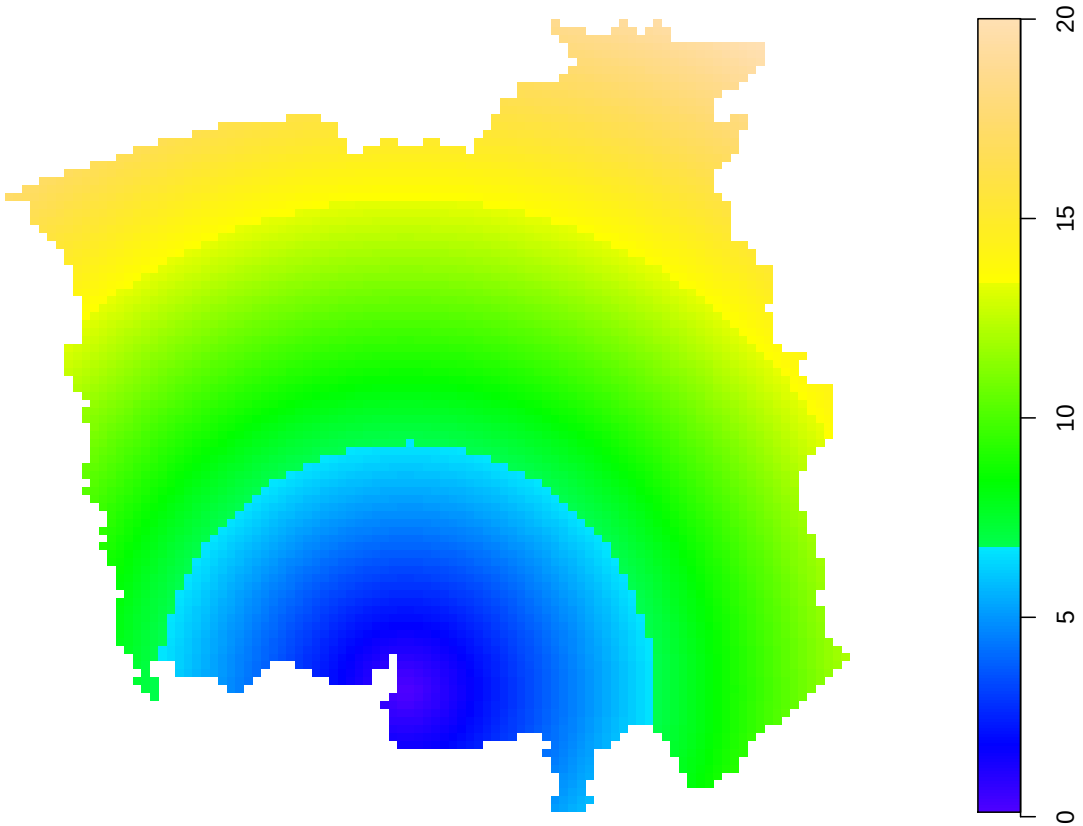
```
> data(chorley)
> chorley
marked planar point pattern: 1036 points
multitype, with levels = larynx       lung
window: polygonal boundary
enclosing rectangle: [343.45, 366.45] x [410.41, 431.79] km
```

Extraction of marks (categorical marks are encoded as a factor):

```
> m <- marks(chorley)
> class(m)
[1] "factor"
> summary(m)
larynx   lung
    58    978
```

Covariate function for cancer cases (incinerator distance)

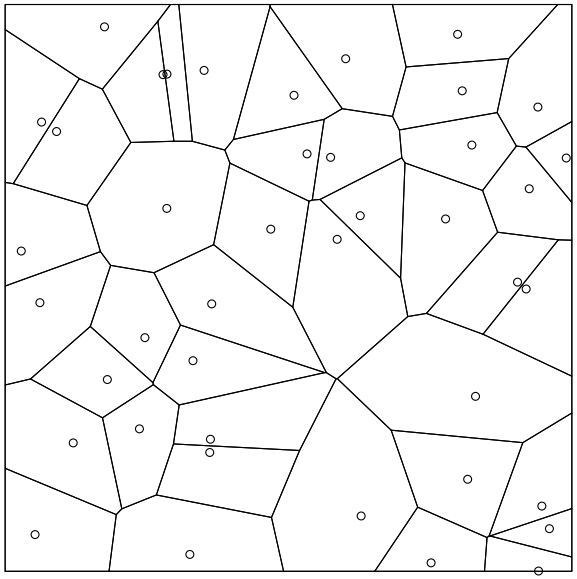
As previously mentioned we can use the incinerator location to define a spatial covariate function at each location $u \in W$ specifying the distance to the incinerator:



Tesselations in spatstat

A "tessellation" is simply a division of space into non-overlapping regions ("tiles"), and is of class `tess` in `spatstat`. The Dirichlet (or Voronoi) tessellation of a point pattern:

```
> X <- runifpoint(42)
> dirichX <- dirichlet(X)
> plot(dirichX, main = "")
> plot(X, add = TRUE)
```

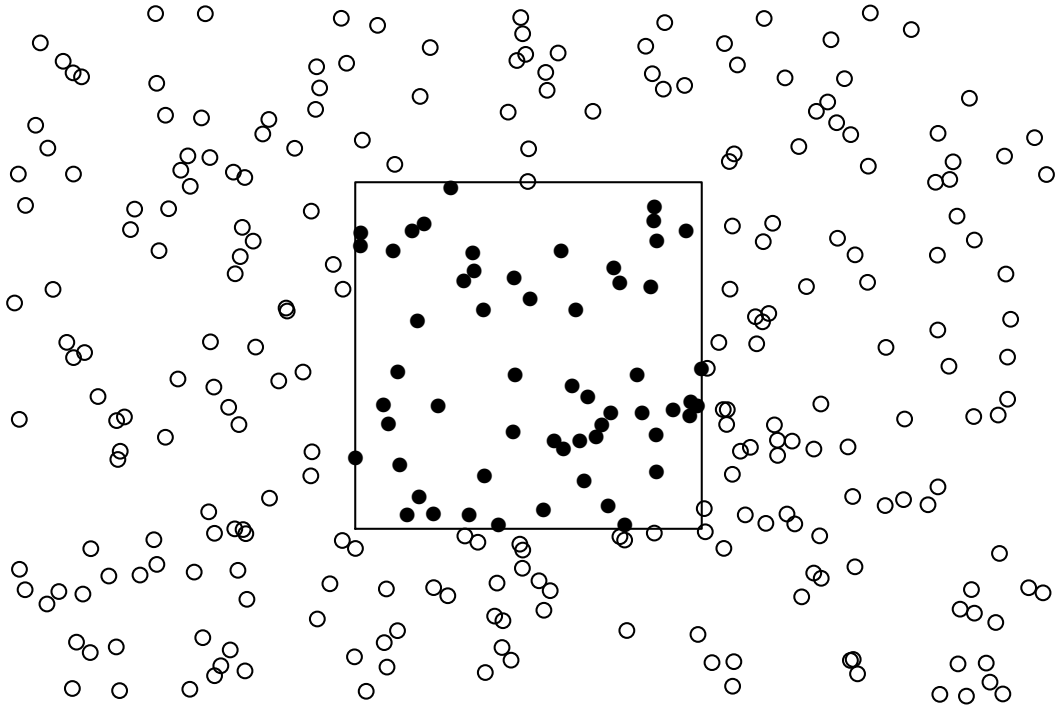


Mathematical model: Point processes

- We treat the observed point pattern $\mathbf{x}$ as a realisation of a random point process $\mathbf{X}$ in two-dimensional space.
- A point process is simply a random set of points; the number of points is random, as well as the locations of the points.
- Our goal is usually to describe the distribution of $\mathbf{X}$ parametrically and estimate the parameters based on an observed point pattern $\mathbf{x}$.
- Two common assumptions for a point process model are *stationarity* and *isotropy* meaning respectively that the distribution of $\mathbf{X}$ is invariant under translation and rotation.

Standard model

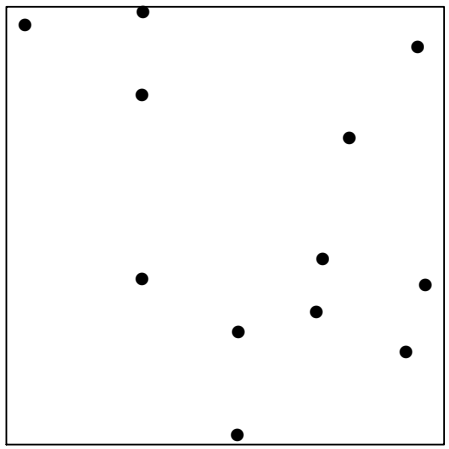
The "standard model" assumes that the point process $\mathbf{X}$ extends throughout 2-D space, but is observed only inside a region W.



It is important to know the window W, since we need to know where points were not observed!

Significance of the absence of points

The significance of knowing where points were *not* observed becomes apparent when thinking of the discretised version of a point process. Suppose the window W is chopped into a large number of tiny 'pixels'. Each pixel is assigned the value $I = 1$ if it contains a point of $\mathbf{X}$, and $I = 0$ otherwise.



1	0	0	1	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0
0	0	0	1	0	0	0	1	0	1
0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0

To investigate the dependence of these indicators on a covariate, we need to observe the covariate value at some locations where $I = 0$, and not only at locations where $I = 1$.

First order properties: The intensity

- For any $B \subset \mathbb{R}^2$ with finite area $|B|$ we define the random variable $N(\mathbf{X} \cap B)$ as the number of points from $\mathbf{X}$ falling in B, and the intensity measure Λ as

$$\Lambda(B) = \mathbb{E}[N(\mathbf{X} \cap B)].$$

- If the intensity measure can be written as

$$\mathbb{E}[N(\mathbf{X} \cap B)] = \int_B \lambda(u)\,\mathrm{d}u$$

  we call $\lambda(\cdot)$ the intensity function.
- If $\lambda(u) = \lambda$ is constant for all u the process is homogeneous. Otherwise it is inhomogeneous.
- A useful formula is

$$\mathbb{E}\left(\sum_{x_i \in \mathbf{X}} h(x_i)\right) = \int h(u)\lambda(u)\,\mathrm{d}u \qquad (1)$$

Intensity estimation

- If the point process is known to be homogeneous an unbiased estimate of the intensity is

$$\overline{\lambda} = \frac{n(\mathbf{x})}{\text{area}(W)}.$$

- This estimate can either be calculated directly or extracted from the summary of a ppp-object:

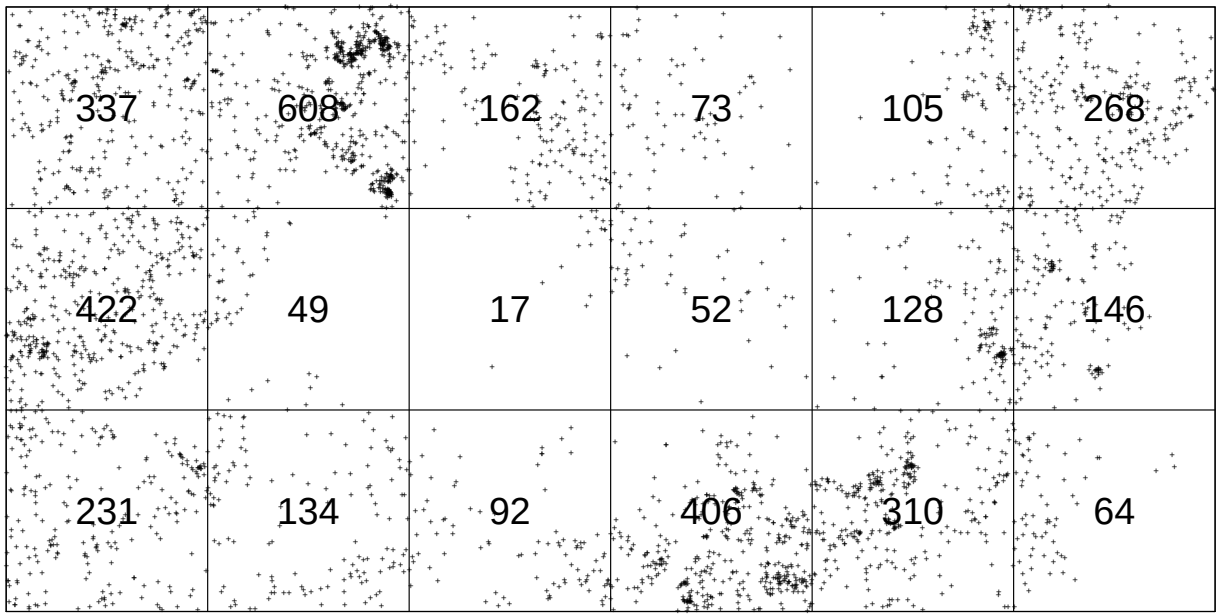
```
> npoints(bei)/area.owin(bei)
[1] 0.007208
> summary(bei)$intensity
[1] 0.007208
```

- In the inhomogeneous case, the intensity function or intensity measure can be estimated non-parametrically by techniques such as quadrat counting and kernel smoothing.

Quadrat counting

The window W is divided into subregions ('quadrats') $B_1, \ldots, B_m$ of equal area. For $j = 1, \ldots, m$ the counts $n_j = n(\mathbf{x} \cap B_j)$ are unbiased estimators of $\Lambda(B_j)$.

```
> Q <- quadratcount(bei, nx = 6, ny = 3)
> plot(bei, cex = 0.5, pch = "+", main = "")
> plot(Q, add = TRUE, cex = 2)
```



Kernel smoothing

The usual *kernel estimator* of the intensity function is

$$\widetilde{\lambda}(u) = e(u) \sum_{i=1}^{n} \kappa(u - x_i),$$

where $\kappa(u)$ is the kernel (an arbitrary probability density) and

$$e(u)^{-1} = \int_W \kappa(u - v)\, \mathrm{d}v$$

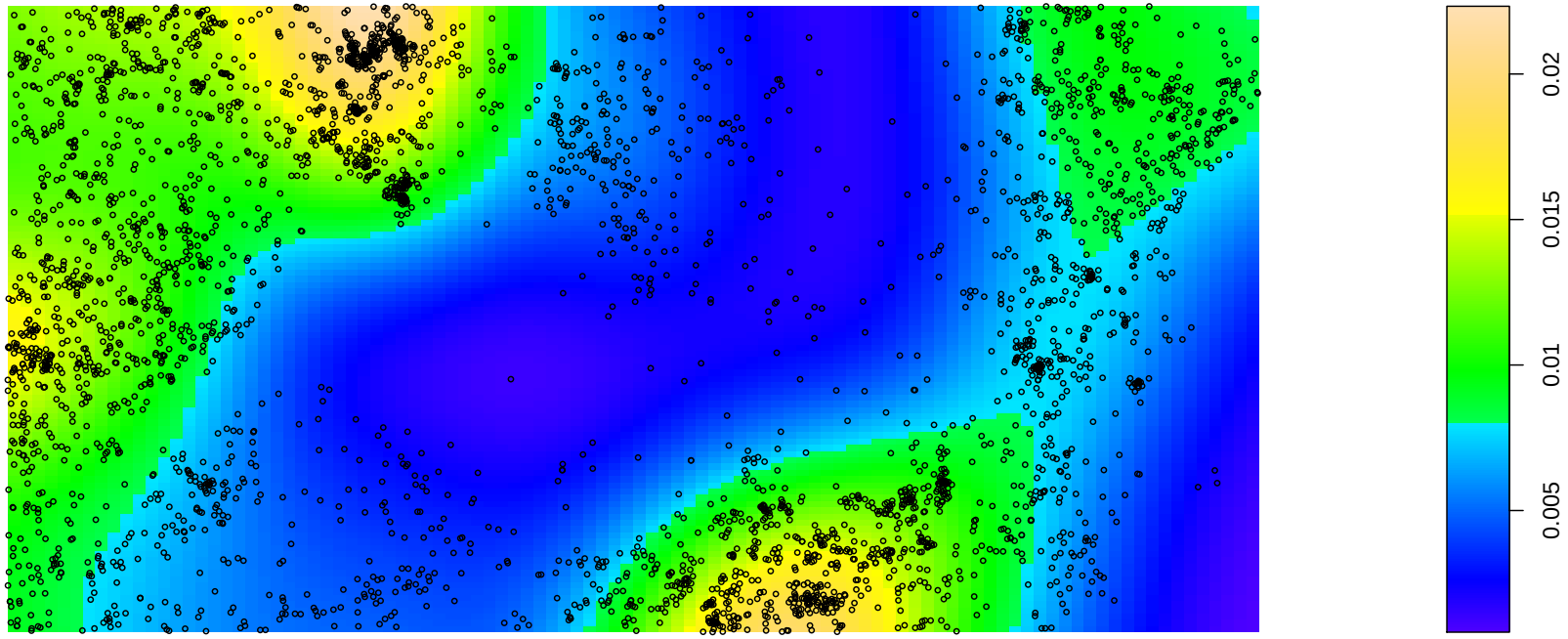
is an edge effect bias correction. By (1) we see that $\widetilde{\lambda}(u)$ is an unbiased estimator of

$$\lambda^*(u) = e(u) \int_W \kappa(u - v)\lambda(v)\, \mathrm{d}v,$$

a smoothed version of the true intensity function $\lambda(u)$.

Kernel smoothing in spatstat

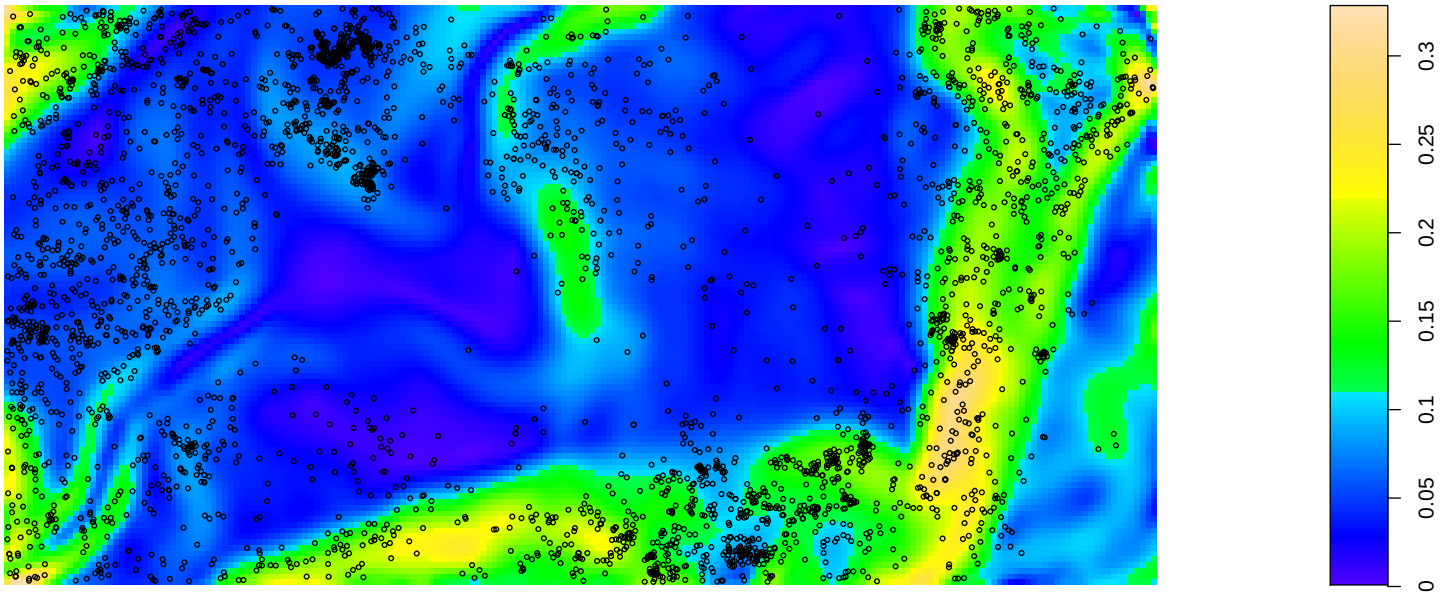
```
> den <- density(bei, sigma = 70)
> plot(den, main = "")
> plot(bei, add = TRUE, cex = 0.5)
```



Dependence of intensity on a covariate

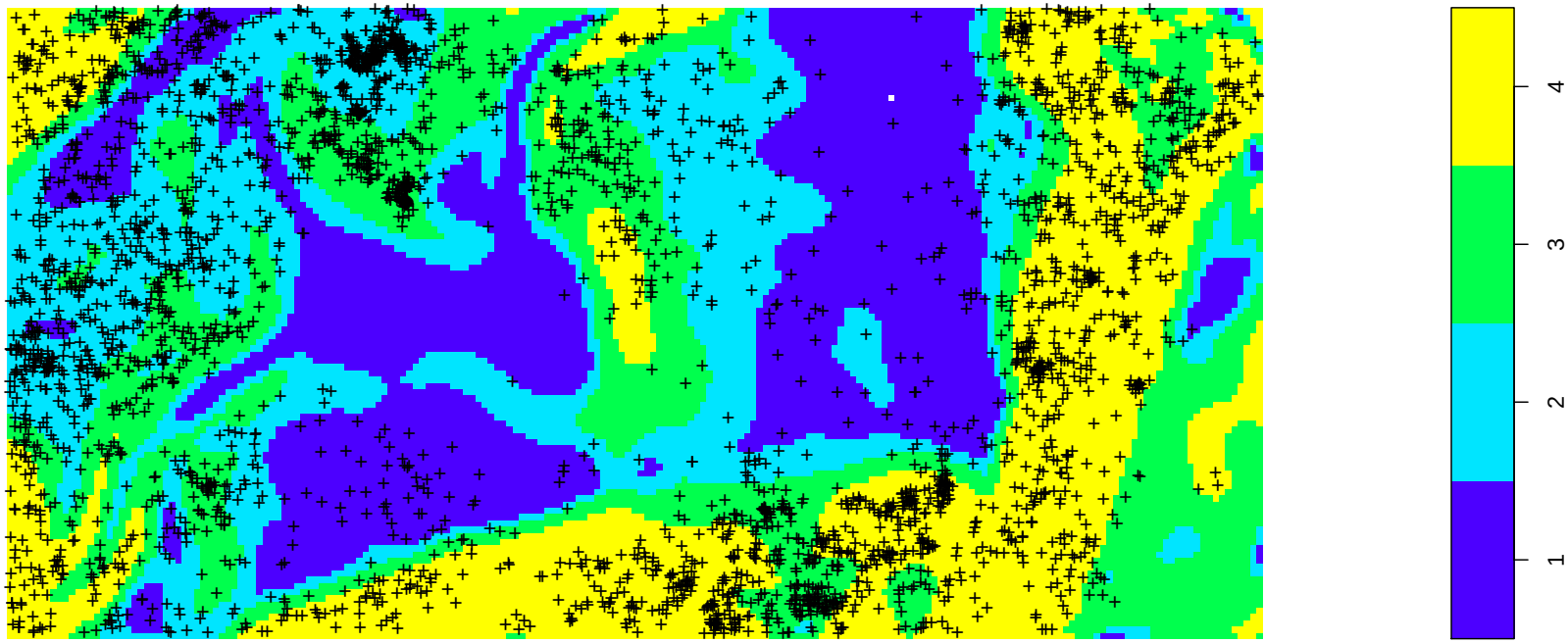
A typical scientific question of interest is whether a covariate effects the abundance of points. For example: Do the rainforest trees prefer steep or flat terrain?

```
> slope <- bei.extra$grad
> plot(slope, main = "")
> plot(bei, add = TRUE, cex = 0.5)
```



Dependence of intensity on a covariate

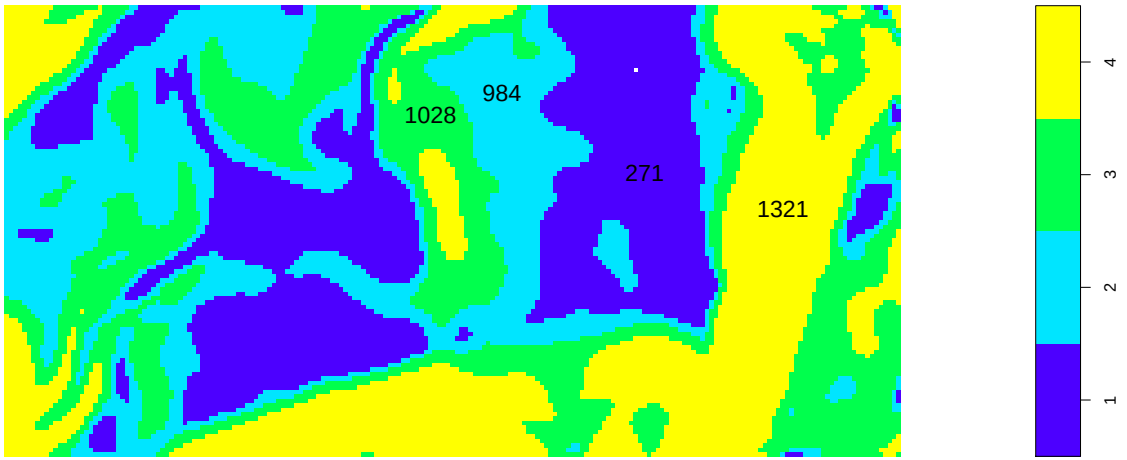
```
> b <- quantile(slope, probs = (0:4)/4)
> slopecut <- cut(slope, breaks = b, labels = 1:4)
> V <- tess(image = slopecut)
> plot(V, main = "")
> plot(bei, add = TRUE, pch = "+")
```



Dependence of intensity on a covariate

Since the four regions have equal area, the counts should be approximately equal if there is a uniform density of trees.

```
> qb <- quadratcount(bei, tess = V)
> qb
tile
   1    2    3    4
 271  984 1028 1321
> plot(qb, main = "", cex = 1.4)
```



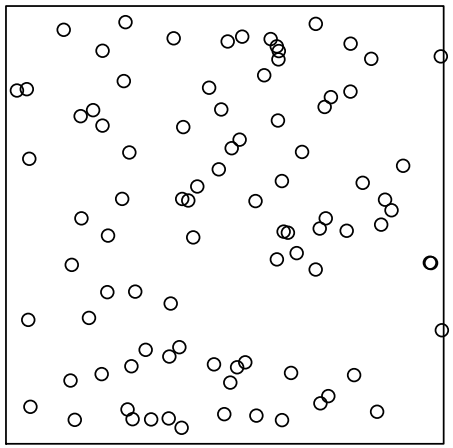
Homogeneous Poisson processes

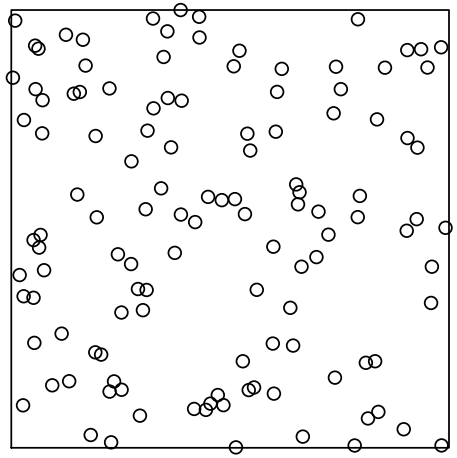
- The homogeneous Poisson process with intensity λ is defined by two properties:
  - $N(A)$ is Poisson distributed with mean $\lambda|A|$, for all A.
  - Conditional on $N(A) = n$ the n points are independent and uniformly distributed in A.
- The counts $N(A)$ and $N(B)$ are independent for disjoint A, B.
- Model for "Complete Spatial Randomness" (CSR).
- Often serves as the null model in a statistical analysis.

Homogeneous Poisson processes in spatstat

A realization of the homogeneous Poisson process with intensity $\lambda = 100$ in the unit square can be generated in two equivalent ways:

```
> plot(rpoispp(100), main = "")
> m <- rpois(1, 100)
> plot(runifpoint(m), main = "")
```





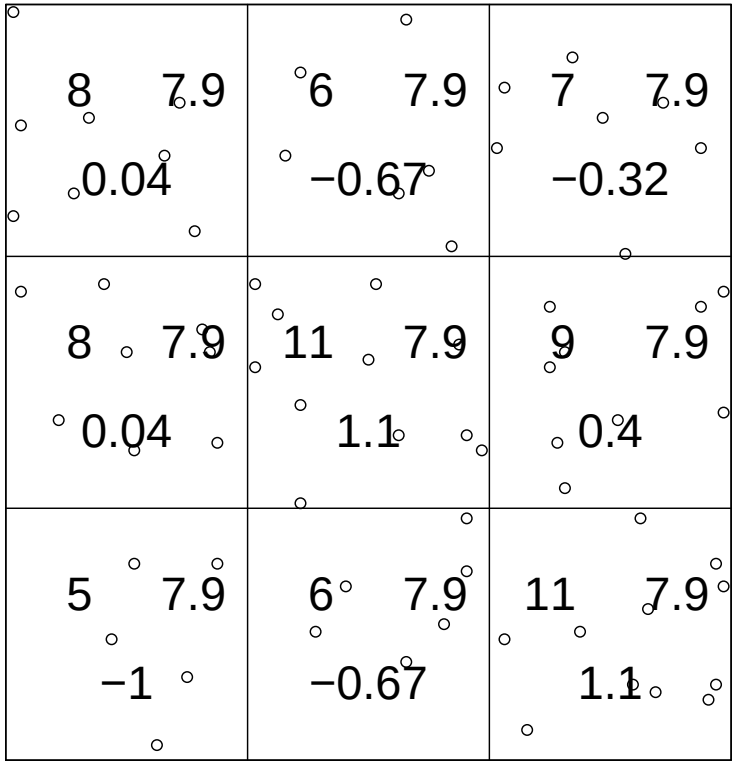
Quadrat counting test for CSR

- Often, an early step in the analysis is to find evidence *against* CSR, which can be done by quadrat counting.
- The window W is divided into subregions ('quadrats') $B_1, \ldots, B_m$ of equal area.
- Under the null hypothesis of CSR, $N(\mathbf{X} \cap B_j)$ are i.i.d. Poisson random variables with the same expected value, so the Pearson χ^2 goodness-of-fit test can be used:

```
> data(swedishpines)
> swetest <- quadrat.test(swedishpines, nx = 3,
+     ny = 3)
> swetest

        Chi-squared test of CSR using quadrat counts

data:  swedishpines
X-squared = 4.6761, df = 8, p-value = 0.7916
```

Quadrat counting test for CSR

```
> plot(swedishpines, main = "")
> plot(swetest, add = TRUE, cex = 2)
```



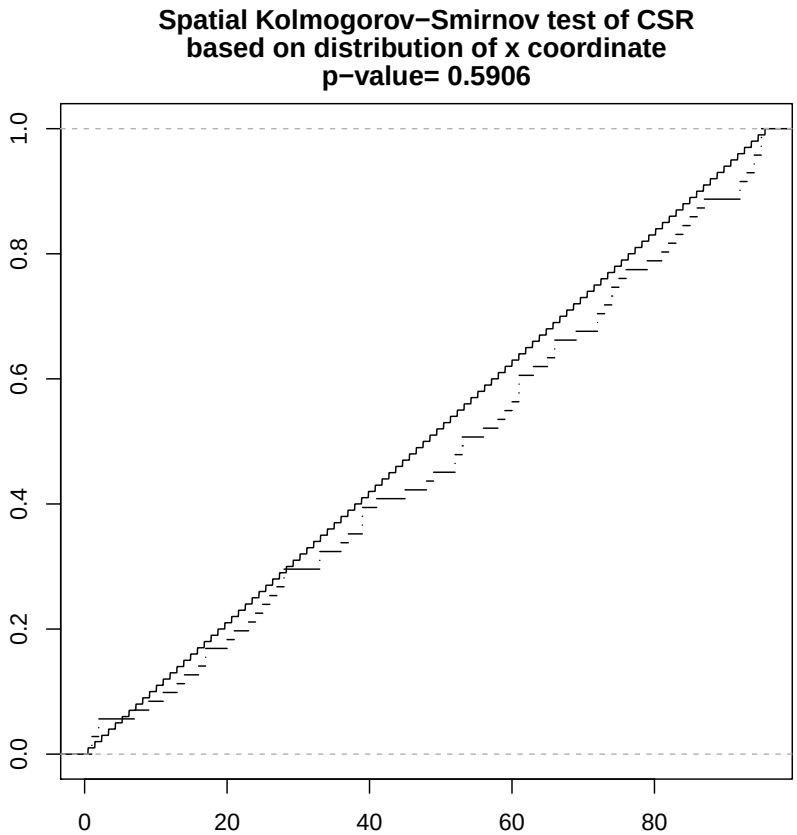
Kolmogorov-Smirnov test for CSR

- Typically a more powerful test of CSR is the Kolmogorov-Smirnov test in which we compare the observed and expected distributions of the values of some function T.
- We specify a real-valued function $T(x, y)$ defined at all locations (x, y) in the window. We evaluate this function at each of the data points. Then we compare this empirical distribution of values of T with the predicted distribution of values of T under CSR, using the classical Kolmogorov-Smirnov test.

Kolmogorov-Smirnov test for CSR

Example with the function $T(x, y) = x$:

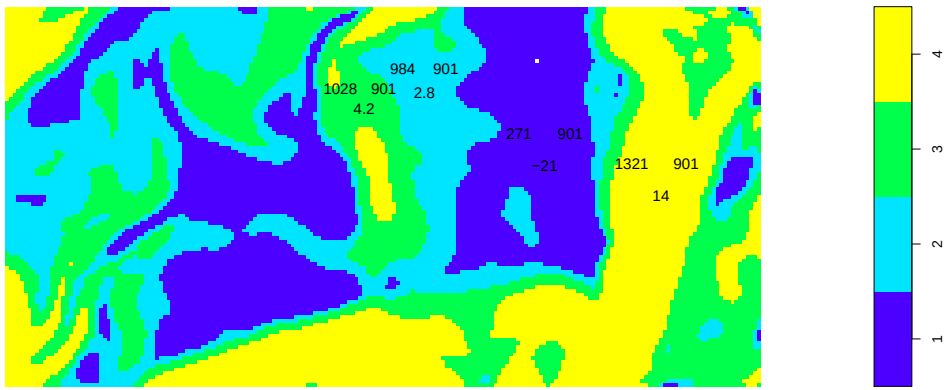
```
> X <- swedishpines
> swetest <- kstest(X, "x")
> plot(swetest)
```



Using covariate data in CSR tests

For the rainforest data `V` is still the tesselation with four areas based on slope as before:

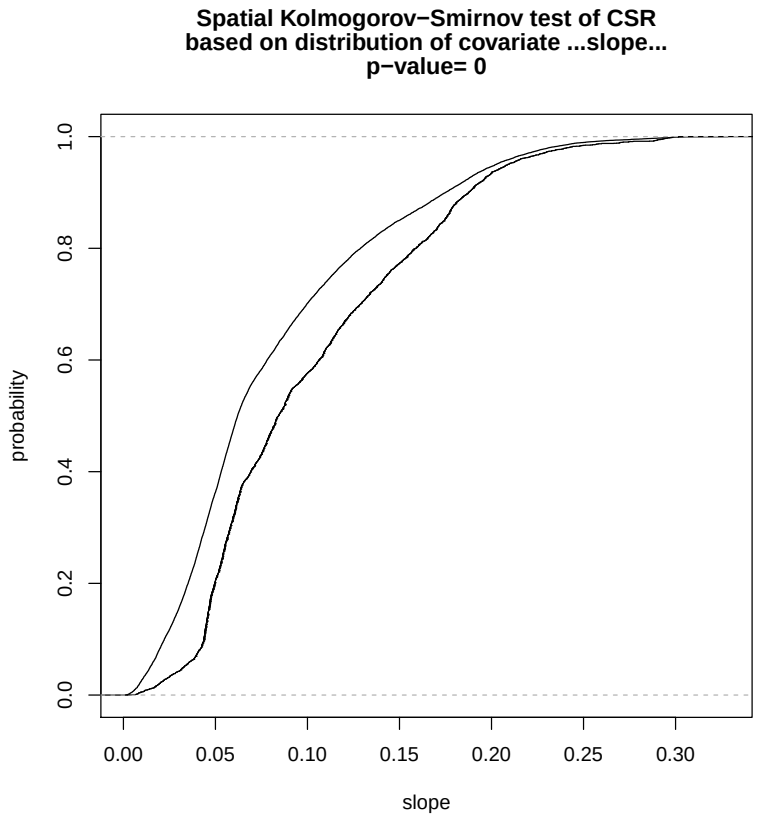
```
> beitest <- quadrat.test(bei, tess = V)
> beitest

        Chi-squared test of CSR using quadrat counts


data:  bei
X-squared = 661.8402, df = 3, p-value < 2.2e-16
> plot(beitest, main = "", cex = 1.1)
```



Using covariate data in CSR tests

Alternatively a Kolmogorov-Smirnov test can be used without making a tesselation:

```
> plot(kstest(bei, slope))
```



Time for a break!