Using Spatstat for Density-based Point Pattern Analysis

1. With the spatstat library loaded, let's look at how we can exploit functions within the package to explore spatial variations in the intensity of a point pattern over a study area.

2. Two primary ways we examine variations in the intensity of a point pattern are with kernel density estimates and with quadrat analysis.

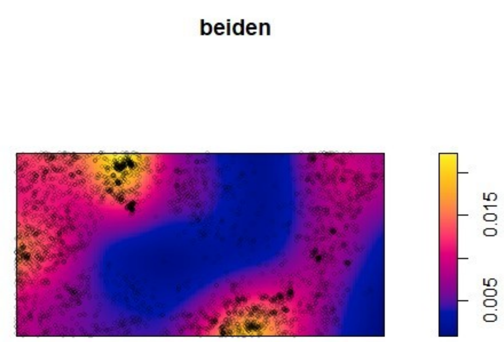
3. Let's examine the point pattern in the data set bei. These data show the coordinates of trees in a tropical rainforest.

4. Load the ppp labelled bei and use the summary function to find the average intensity of points over the study region

```
> data(bei) > summary(bei)
Planar point pattern:  3604 points
Average intensity 0.007208 points per square metre
Coordinates are given to 1 decimal place, i.e. rounded to the nearest multiple of 0.1 metres
Window: rectangle = [0, 1000] x [0, 500] metres
Window area = 5e+05 square metres
Unit of length: 1 metre
```

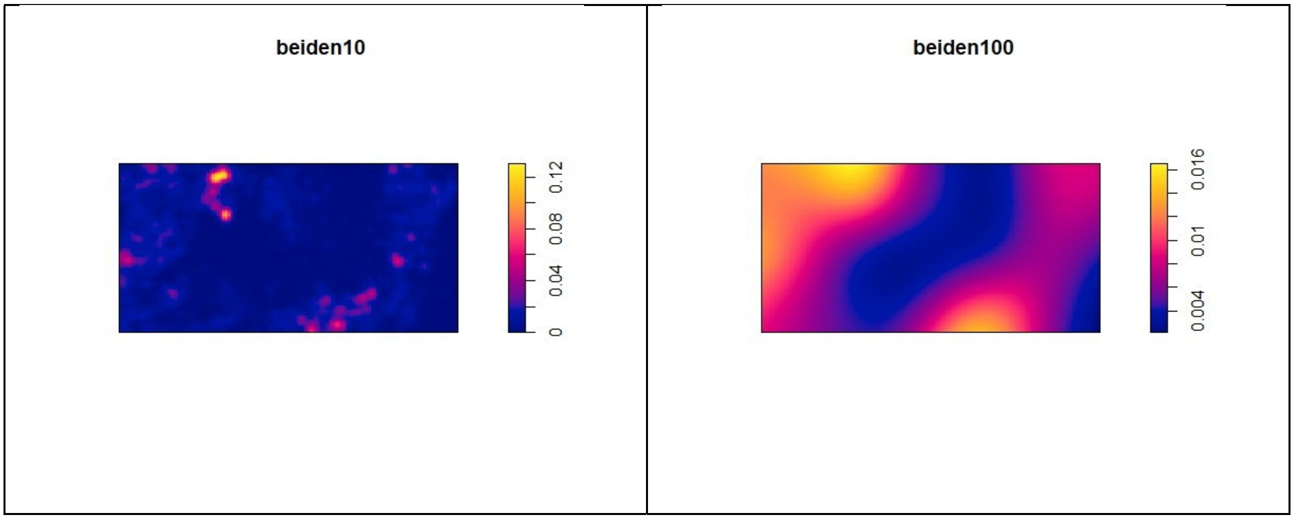
5. Kernel density estimates are generated using the density function in spatstat. The generic command uses a Gaussian kernel (the third line just adds the individual points to the KDE plot)

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



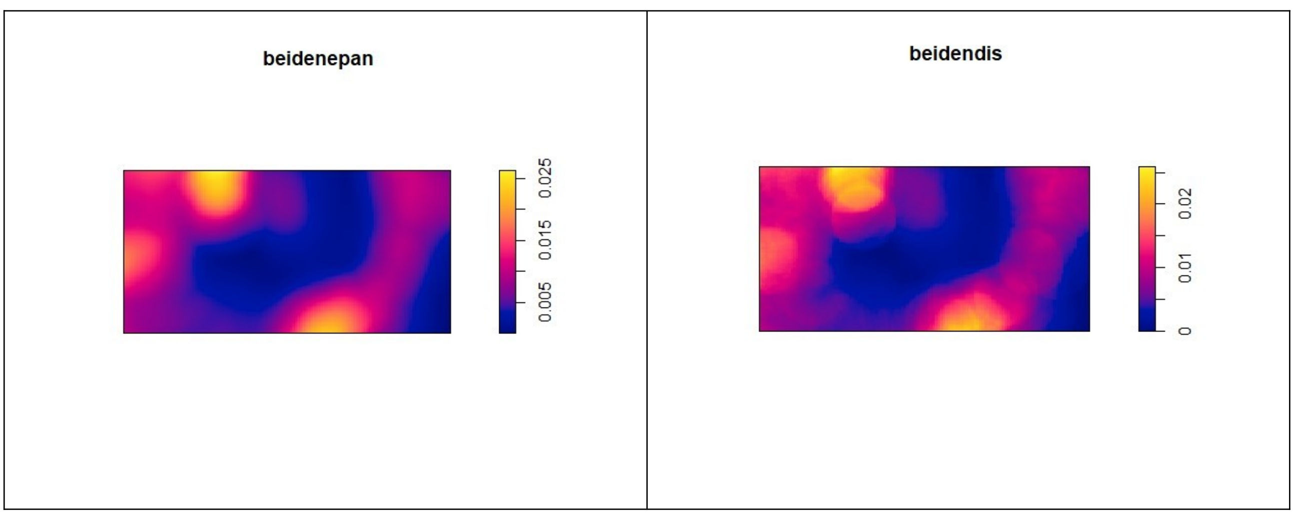
6. Sigma in the kernel density estimation controls the bandwidth (the amount of smoothing) of the density function

> beiden10 <- density(bei, sigma=10) > plot(beiden10) > beiden100 <- density(bei, sigma=100) >plot(beiden100)



7. You can try different kernel types (gaussian(default), epanechnikov, quartic, disc) in the following way

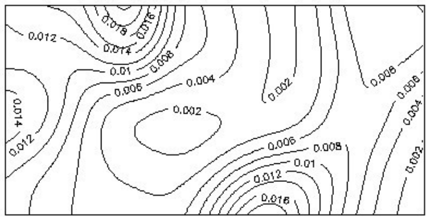
> beidenepan <- density(bei, sigma=50, kernel=c("epanechnikov")) > plot(beidenepan) > beidendis <- density(bei, sigma=50, kernel=c("disc")) > plot(beidendis)



8. You can also generate contour plots of the KDE.

> contour(beiden)

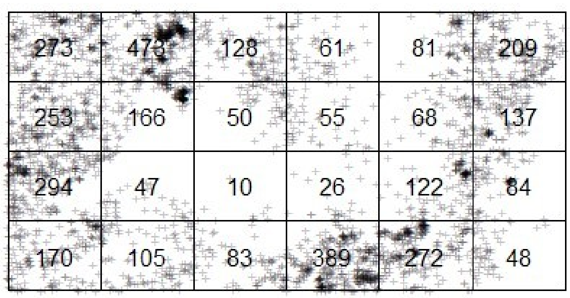
beiden



9. Now let us turn to quadrat analysis of the bei point pattern. Quadrat counting uses the quadratcount function. So to count across 24 (6x4) quadrats of the bei data

```
> beiq <- quadratcount(bei, nx=6, ny=4) > plot(bei, cex=0.75, pch="+") > plot(beiq, add=TRUE, cex=1.0)
```

bei



10. What about a quick hypothesis test for quadrat count data? We typically assume a null hypothesis that the point pattern was produced by a homogeneous Poisson process that is consistent with Complete Spatial Randomness (CSR), ie that is IRP. The standard test in spatstat generates a test statistic for a chi-squared distribution.

11. We can perform the test as

```
> quadrat.test(bei, nx=6, ny=4)
```

Chi-squared test of CSR using quadrat counts data:
bei
X2 = 2222, df = 23, p-value < 2.2e-16 and given this p-value, we would reject the null hypothesis
alternative hypothesis: two.sided
Quadrats: 6 by 4 grid of tiles

12. We can also generate the observed and expected count data for the test to save work. In the form of a plot. I do this for only 12 quadrats to ease reading of the data

```
> plot(bei, cex=0.5, pch="+") > beiqtest <-quadrat.test(bei, nx=4, ny=3) > plot(beiqtest, add=TRUE, cex=1.0)
```

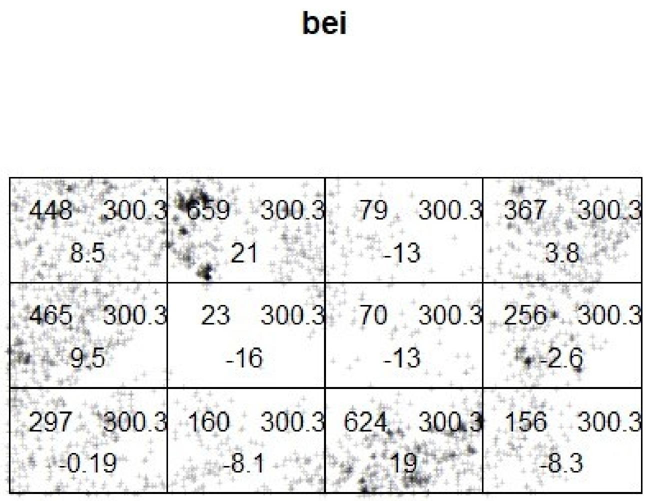
Observed counts are shown in the top-left of the quadrats and expected values in the top-right. The difference (not squared) is shown at the bottom. Simply sum the squared differences to get the numerator of the chi-squared test statistic.

```
> beiqtest
```

Chi-squared test of CSR using quadrat counts

data: bei
X2 = 1692.2, df = 11, p-value < 2.2e-16 alternative
hypothesis: two.sided
Quadrats: 4 by 3 grid of tiles

And the plot and key numbers are shown on the next page


bei
448 300.3
8.5
659 300.3
21
79 300.3
-13
367 300.3
3.8
465 300.3
9.5
23 300.3
-16
70 300.3
-13
256 300.3
-2.6
297 300.3
-0.19
160 300.3
-8.1
624 300.3
19
156 300.3
-8.3