Analysing Spatial Data in R: Worked example: point patterns

Roger Bivand

Department of Economics
Norwegian School of Economics and Business Administration
Bergen, Norway

31 August 2007

Outline

- What we see on a map is a pattern, or perhaps some patterns mixed together.
- It is not easy to work back from map pattern to the process or processes that generated it/them.
- Using a variety of approaches, we can explore and analyse point patterns, also reviewing an important chapter in the development of quantitative geography.
- Practically, we will also see how we can try out different approaches, and how their assumptions affect our conclusions.

References

- David O'Sullivan and David Unwin (2003) *Geographical Information Analysis*, Wiley, chapter 4, plus chapter 3 for the curious;
- Ian Smalley and David Unwin (1968) The formation and shape of drumlins and their distribution and orientation in drumlin fields, *Journal of Glaciology*, 7, pp. 377–390; Alan R. Hill (1973) The distribution of drumlins in County Down, Ireland, *Annals, AAG*, 63 (2). pp. 226–240.
- Human geographers may also like Trevor Bailey and Anthony Gatrell (1995) *Interactive spatial data analysis*, Longman, chapter 3.

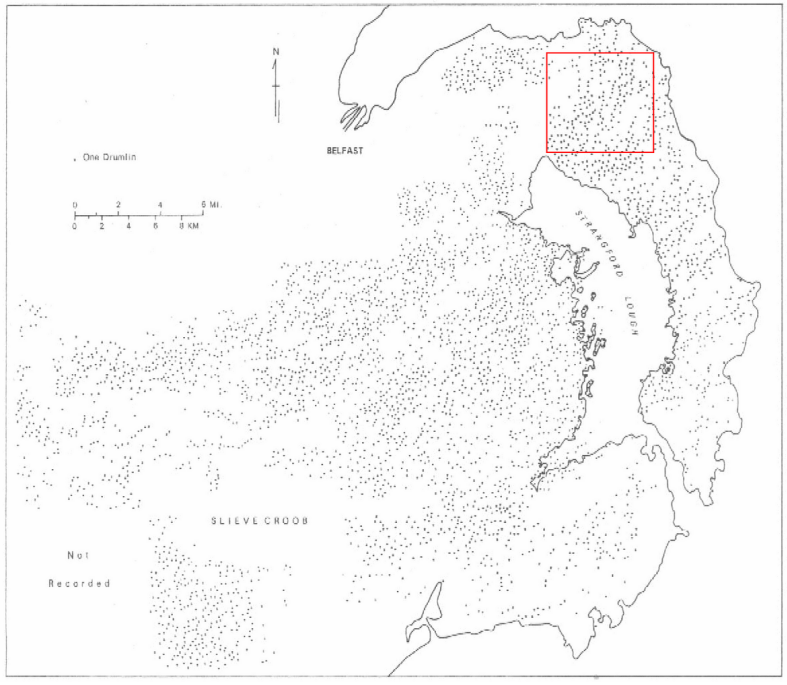
Drumlins, Poland

Data

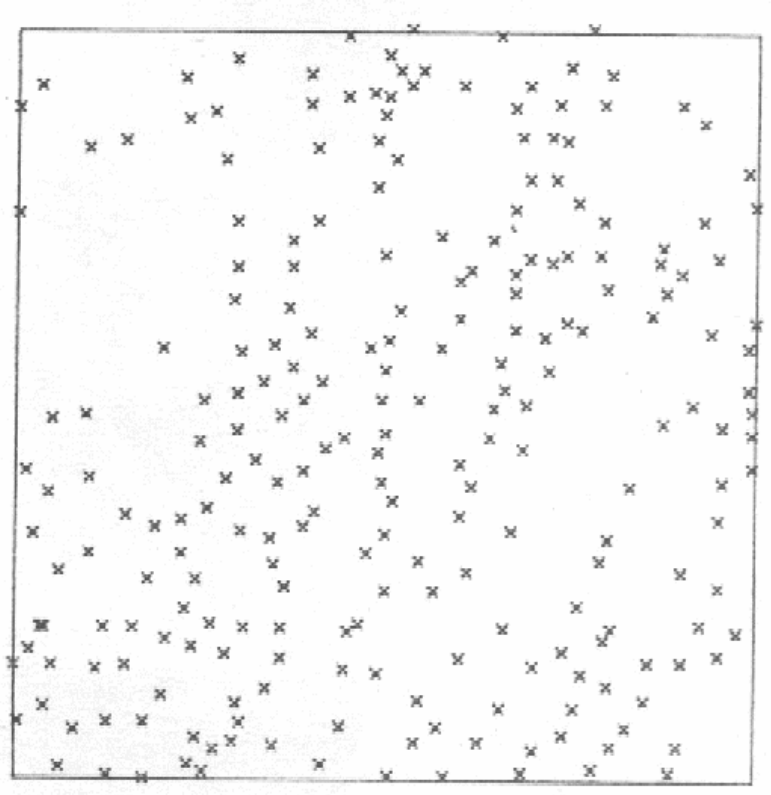
```
> library(rgdal)
> drumlins <- readOGR(".", "drumlins")

OGR data source with driver: ESRI Shapefile
Source: ".", layer: "drumlins"
with  232  rows and  4  columns

> drumlins_SP <- as(drumlins,
+     "SpatialPoints")
```

A data set similar to the one refered to by O'Sullivan and Unwin on p. 100-101 is available in **spatial** in R (associated with Venables and Ripley (2002) Modern Applied Statistics with S) — it is the one used by Upton and Fingleton, coded by Ripley. We have here copied the points to a shapefile.

Drumlins, County Down, Ireland



(Hill, 1973)



(Upton & Fingleton, 1985)

Using **spatstat** with **sp**

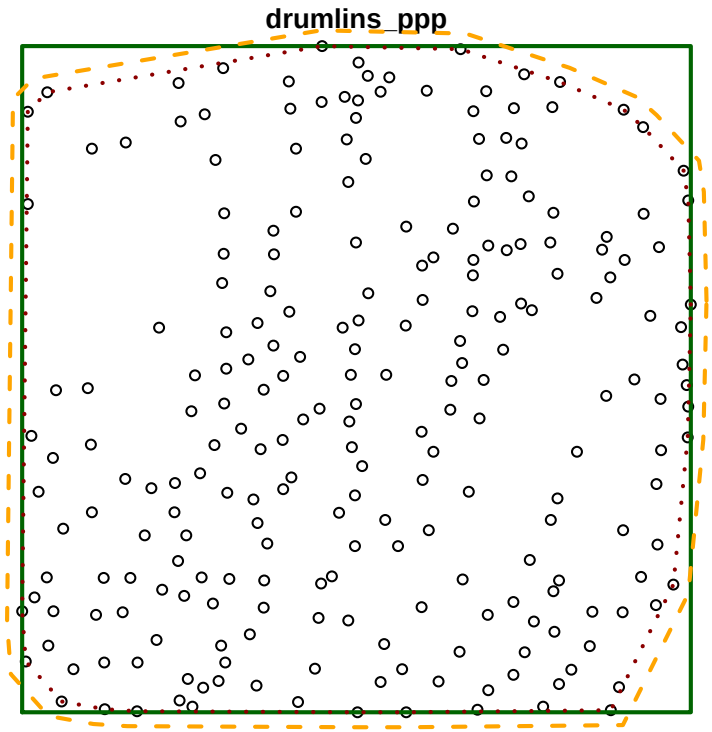
```
> library(maptools)
> library(spatstat)
> drumlins_ppp <- as(drumlins_SP,
+     "ppp")

> drumlins_ppp

planar point pattern: 232
  points
  window: rectangle = [0.28049,
  8.2317] x [5.5671, 13.488]
  units
```

Although **spatstat** and the **sp** classes have developed independently, they have a good deal in common, and point patterns, images and polygon windows can be exchanged

Edges and plot

Point pattern objects need bounding windows to show where the population of data points were collected. The default window is the bounding box of the points, but others are available.

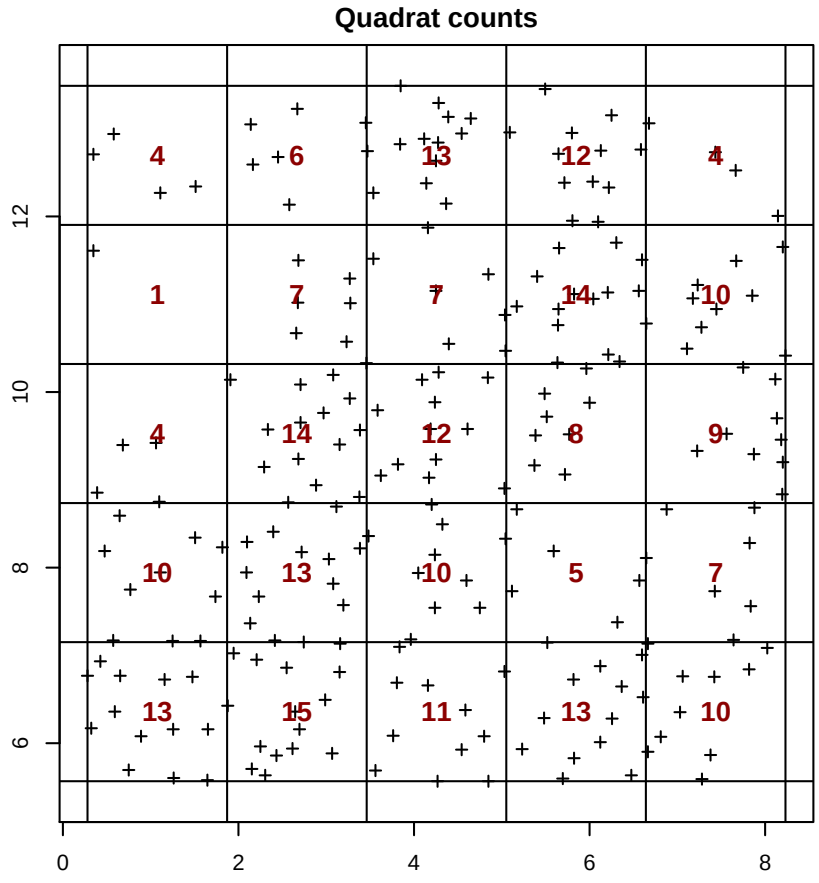
```
> bb <- bounding.box(drumlins_ppp)
> ch <- convexhull.xy(drumlins_ppp)
> rr <- ripras(drumlins_ppp)
> drumlins_rr <- ppp(drumlins_ppp$x,
+     drumlins_ppp$y, window = rr)
```



Quadrat analysis

One legacy approach to point patterns, avoiding the drudge of measuring inter-point distances, has been to divide the study area into quadrats, and count the numbers of points falling into each quadrat. This can take the form of a 2D histogram, or be displayed as an image plot.

```
> qc <- quadratcount(drumlins_ppp)
```



Quadrat tests

Chi-squared tests for Complete Spatial Randomness using quadrat counts may seem attractive, but suffer from the same problems as do histogram bins:

```
> quadrat.test(drumlins_ppp)

Chi-squared test of CSR using
quadrat counts

data:  drumlins_ppp
X-squared = 37.8276, df = 24,
p-value = 0.03611
```

Just adding one more row and column of quadrats, or switching windows, changes our conclusion:

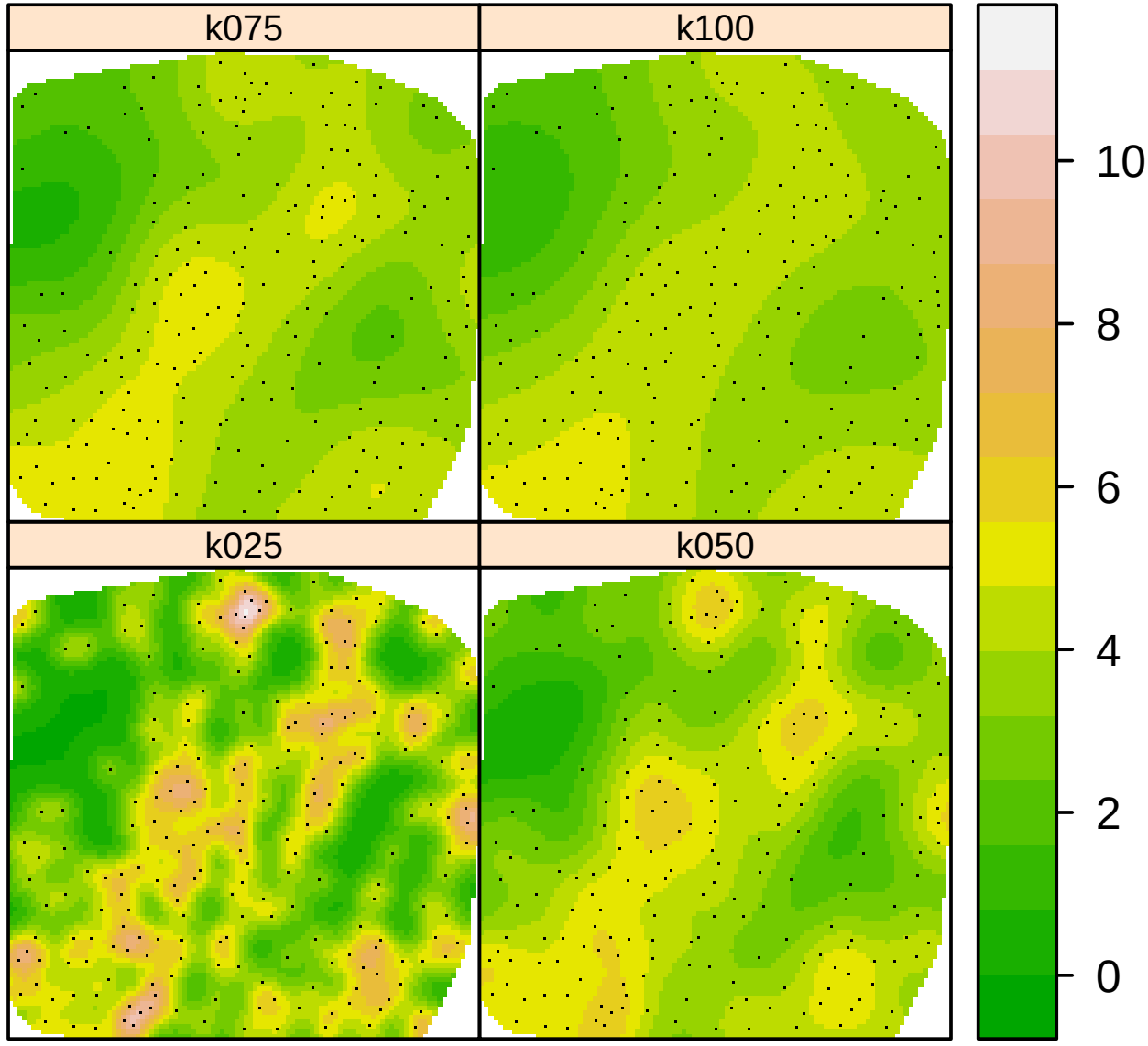
```
> quadrat.test(drumlins_ppp, nx = 6)

Chi-squared test of CSR using
quadrat counts

data:  drumlins_ppp
X-squared = 41.4138, df = 35,
p-value = 0.2110

> quadrat.test(drumlins_rr)

Chi-squared test of CSR using
quadrat counts

data:  drumlins_rr
X-squared = 34.1928, df = 24,
p-value = 0.0813
```

Density plots

Density plots use a 2D kernel, in **spatstat** a Gaussian kernel, to create smoothed histograms avoiding the problems of quadrat counts. The key argument to pass to the density method for point patterm objects is `sigma=`, which determines the bandwidth of the kernel. Since we can coerce the image objects output by the method to an **sp** class, we use this to cumulate density values for different values of sigma.

```
> k025 <- density(drumlins_rr, sigma = 0.25, xy = crds)
> SG <- as(k025, "SpatialGridDataFrame")
> k050 <- density(drumlins_rr, sigma = 0.5, xy = crds)
> SG <- cbind(SG, as(k050, "SpatialGridDataFrame"))
> k075 <- density(drumlins_rr, sigma = 0.75, xy = crds)
> SG <- cbind(SG, as(k075, "SpatialGridDataFrame"))
> k100 <- density(drumlins_rr, sigma = 1, xy = crds)
> SG <- cbind(SG, as(k100, "SpatialGridDataFrame"))
> names(SG) <- c("k025", "k050", "k075", "k100")
```

Kernel density surfaces



Impact of bandwidth on surface

Narrower bandwidths yield more extreme values, broader bandwidths narrow the interquartile range. From this table, we can see how the change in the bandwidth is affecting the relative differences in our view of the local estimates of intensity.

```
> summary(as(SG, "data.frame")[, 1:4])
```

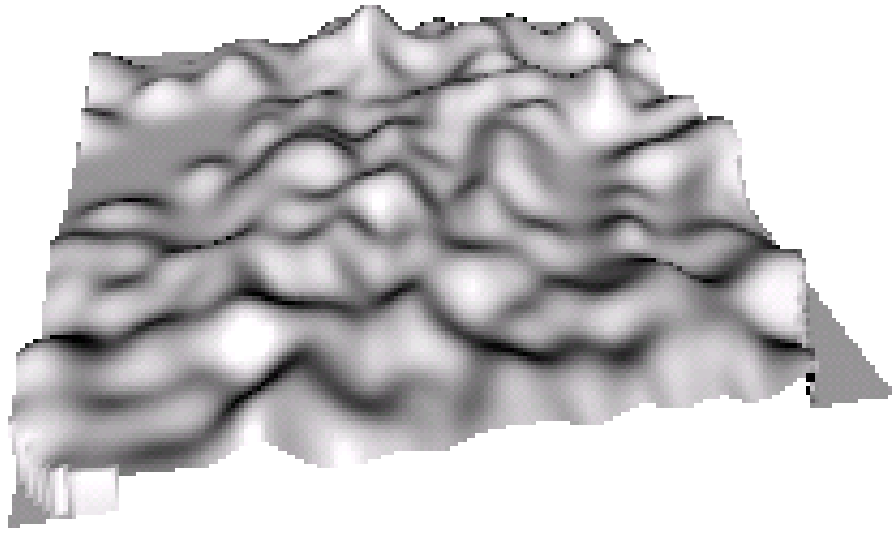
```
      k025                k050              k075              k100
 Min.   :2.463e-04   Min.   :0.1861   Min.   :0.5834   Min.   :0.949
 1st Qu.:2.131e+00   1st Qu.:2.7316   1st Qu.:3.0173   1st Qu.:3.191
 Median :3.632e+00   Median :3.7654   Median :3.7958   Median :3.800
 Mean   :3.640e+00   Mean   :3.6195   Mean   :3.6046   Mean   :3.601
 3rd Qu.:5.105e+00   3rd Qu.:4.7056   3rd Qu.:4.4354   3rd Qu.:4.234
 Max.   :1.114e+01   Max.   :6.1798   Max.   :5.3341   Max.   :5.103
```

Interactive perspective plots

We can also display the local estimates of point intensity as a quasi-3D surface using the **rgl** package, and examine it interactively. This can be manipulated by changing the viewpoint with θ and ϕ, and equivalent parameters for the lighting source. The vertical scale is that of the local intensity.

```
> library(rgl)
> z <- as(SG["k025"], "matrix")
> z[is.na(z)] <- 0
> x <- 1:nrow(z)
> y <- 1:ncol(z)
> rgl.bg(color = "white")
> rgl.clear()
> rgl.viewpoint(theta = 350, phi = 35)
> rgl.surface(x, y, z)

> rgl.snapshot("k025.png")
```

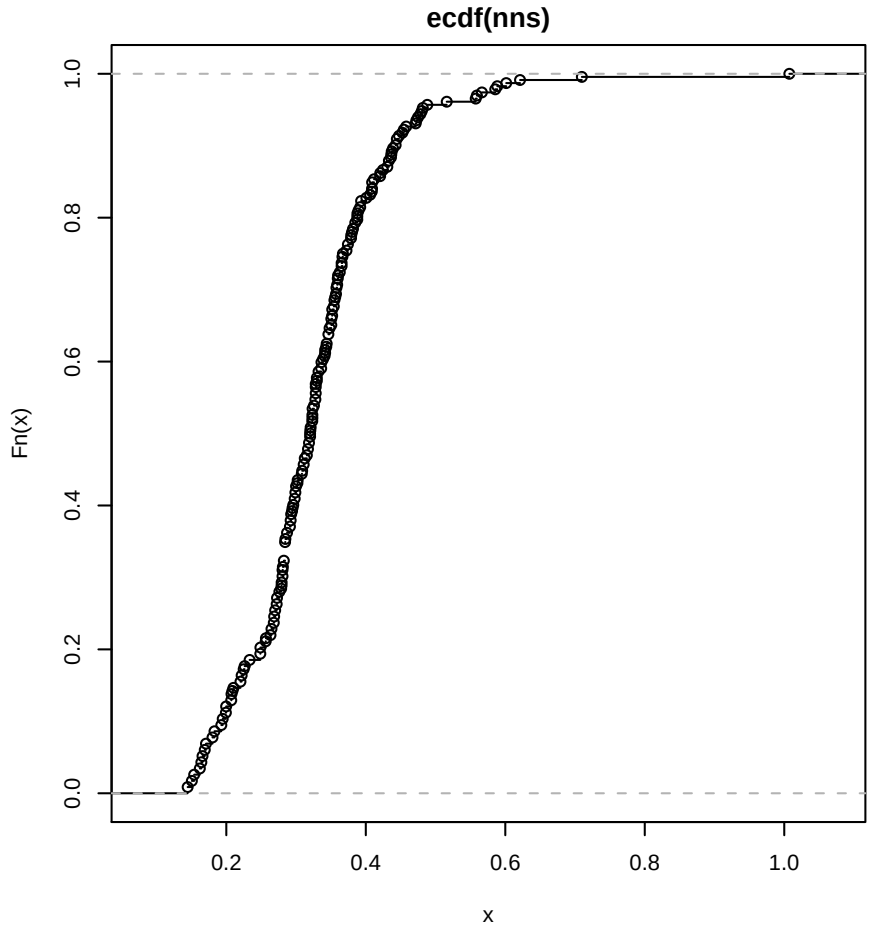
Perspective plot (0.25 bandwidth)



Nearest-neighbour distances

We can find and plot nearest neighbour distances, finding them with `nndist` — plotting the empirical cumulative distribution function of the nearest neighbour distances is interesting:

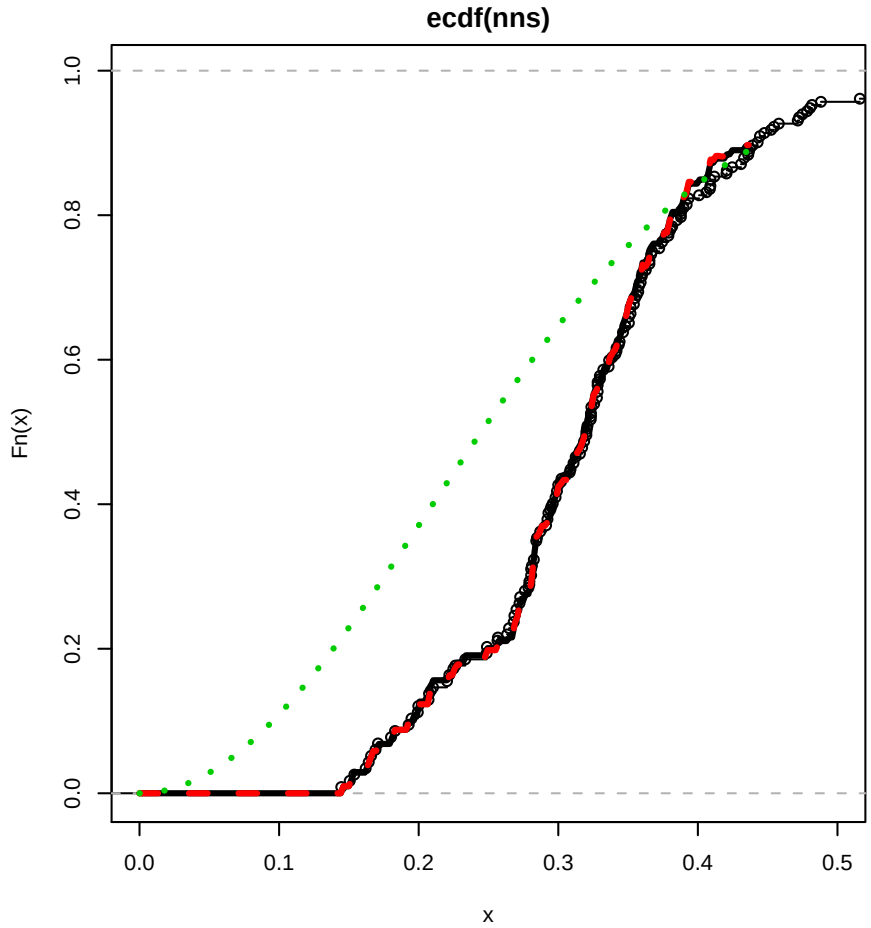
```
> nns <- nndist(drumlins_rr)
> summary(nns)

   Min. 1st Qu.  Median    Mean
 0.1446  0.2700  0.3203  0.3254
3rd Qu.    Max.
 0.3685  1.0070

> plot(ecdf(nns))
```



Using $\hat{G}$ — empirical cumulative distribution function

The $\hat{G}$ measure turns out to be just the ECDF of the nearest neighbour distances, plotted by default with the expected CSR line; `Gest` returns binned values for a range of distance bins best chosen by the function:

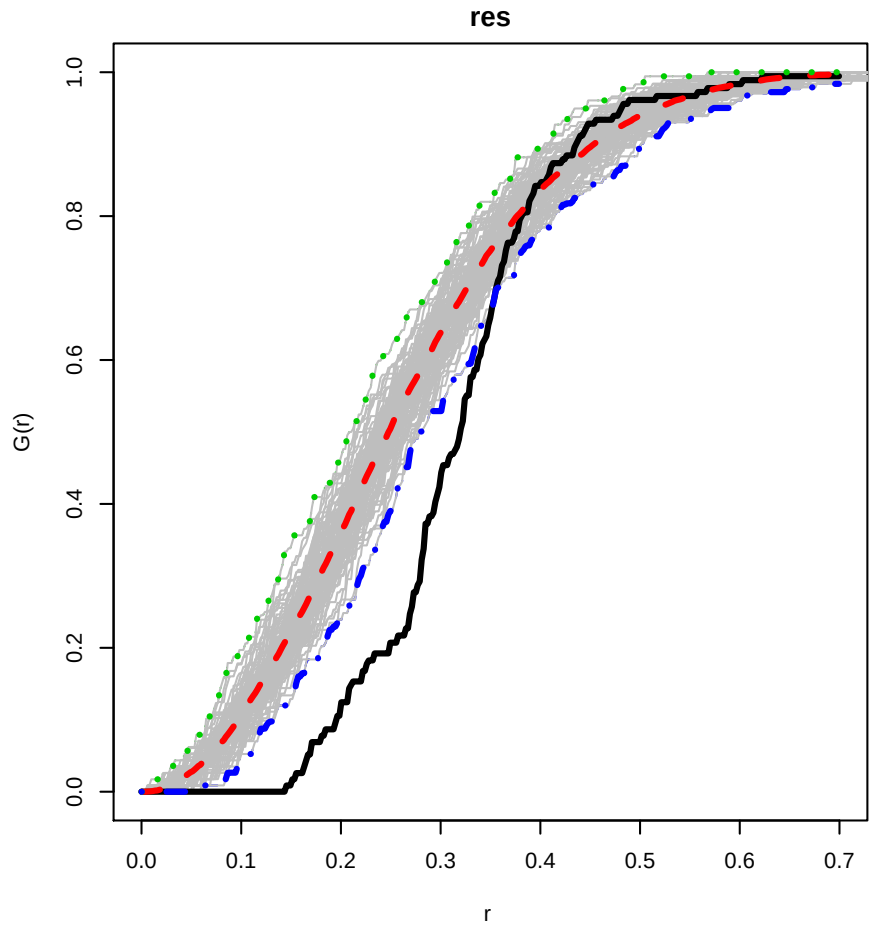
```
> plot(ecdf(nns), xlim = c(0,
+     0.5))
> plot(Gest(drumlins_ppp), add = TRUE,
+     lwd = 3)
```



Simulating $\hat{G}$ for CSR

If we generate many simulated CSR point patterns for the current window, we can use the `envelope` method to explore whether the observed $\hat{G}$ measures lie in relation to the simulated ones:

```
> n <- drumlins_rr$n
> ex <- expression(runifpoint(n,
+     win = rr))
> res <- envelope(drumlins_rr,
+     Gest, nsim = 99, simulate = ex,
+     verbose = FALSE, saveall = TRUE)

> plot(res, xlim = c(0, 0.7))
> for (i in 2:100) lines(attr(res,
+     "savedata")[[1]], attr(res,
+     "savedata")[[i]], col = "grey")
> plot(res, add = TRUE, xlim = c(0,
+     0.7))
```



Clark/Evans *R* statistics

We can also compute the nearest neighbour based Clark/Evans *R* statistic :

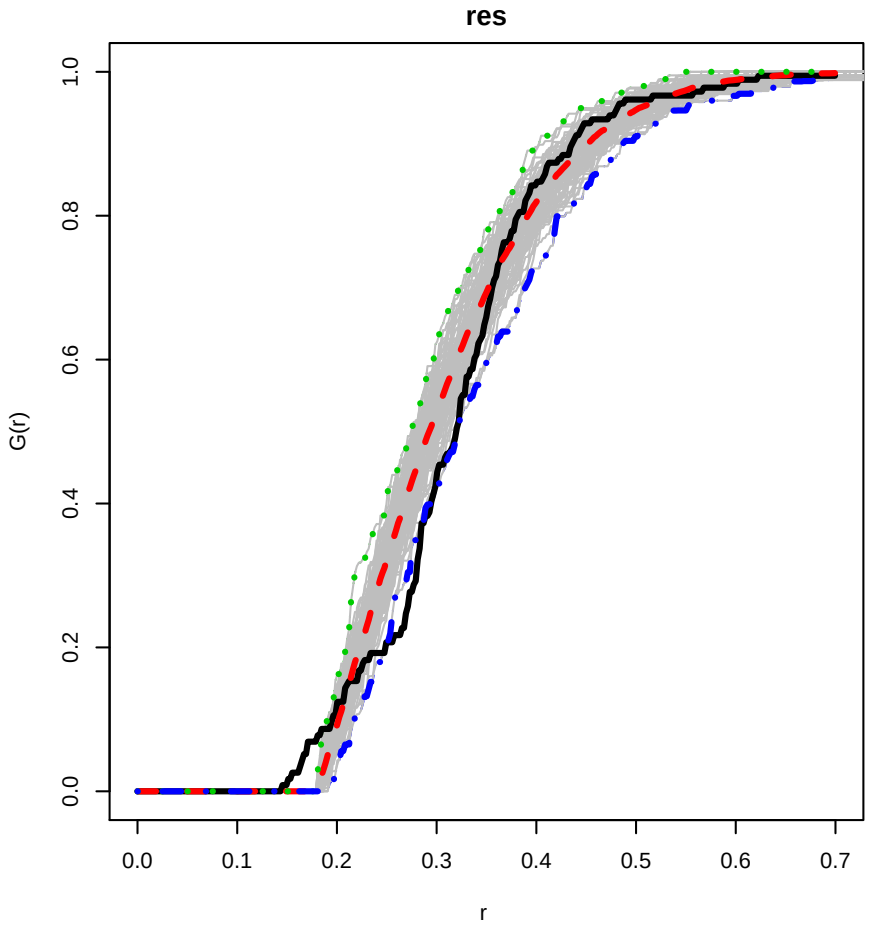
```
> source("clarkevans_rev.R")
> clarkevans(drumlins_ppp)

naive.max  edge.max
 1.248991  1.214479

> clarkevans(drumlins_rr, correction = "none")

   naive
1.238292

> clarkevans(drumlins_rr, correction = "guard", clipregion = erode.owin(rr,
+     r = 1))

   guard
1.199609
```

which seem to indicate that the observed and CSR expected distances are similar, but perhaps more evenly spaced than clustered.

Was CSR a good idea?

From what we have seen, it appears the the drumlin summit points are more regularly than randomly distributed. If we think, however, the absence of short nearest neighbour distance may mean that they pusheach other apart (in fact this is about points not being a good way of representing ellipses) — so we can try to simulate from a Simple Sequential Inhibition (SSI) process with a 180m inhibition radius instead of CSR:

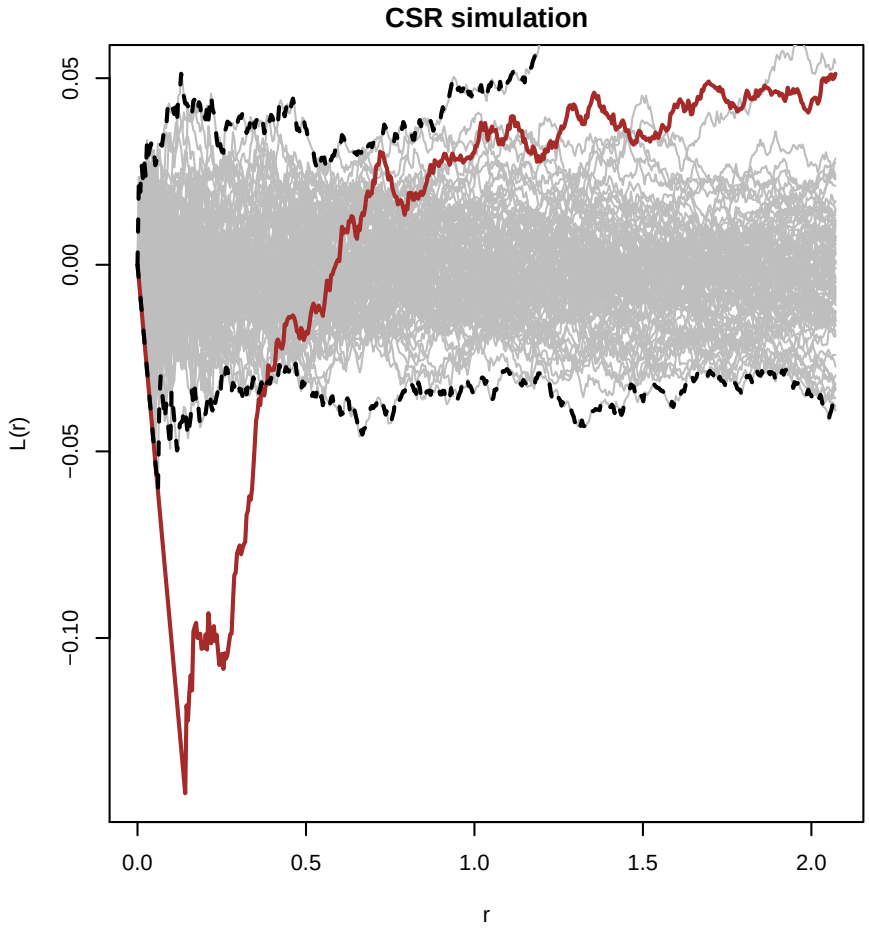
```
> ex <- expression(rSSI(0.18,
+     n, win = rr))
> res <- envelope(drumlins_rr,
+     Gest, nsim = 99, simulate = ex,
+     verbose = FALSE, saveall = TRUE)
```



$\hat{K}$ with CSR simulation

As we know, $\hat{G}$ uses nearest neighbour distances to express summary features of a point pattern. The $\hat{K}$ function uses point intensities in rings spreading out from the points, and so uses more of the data to examine what is driving the process (reported here as $\hat{L}$):

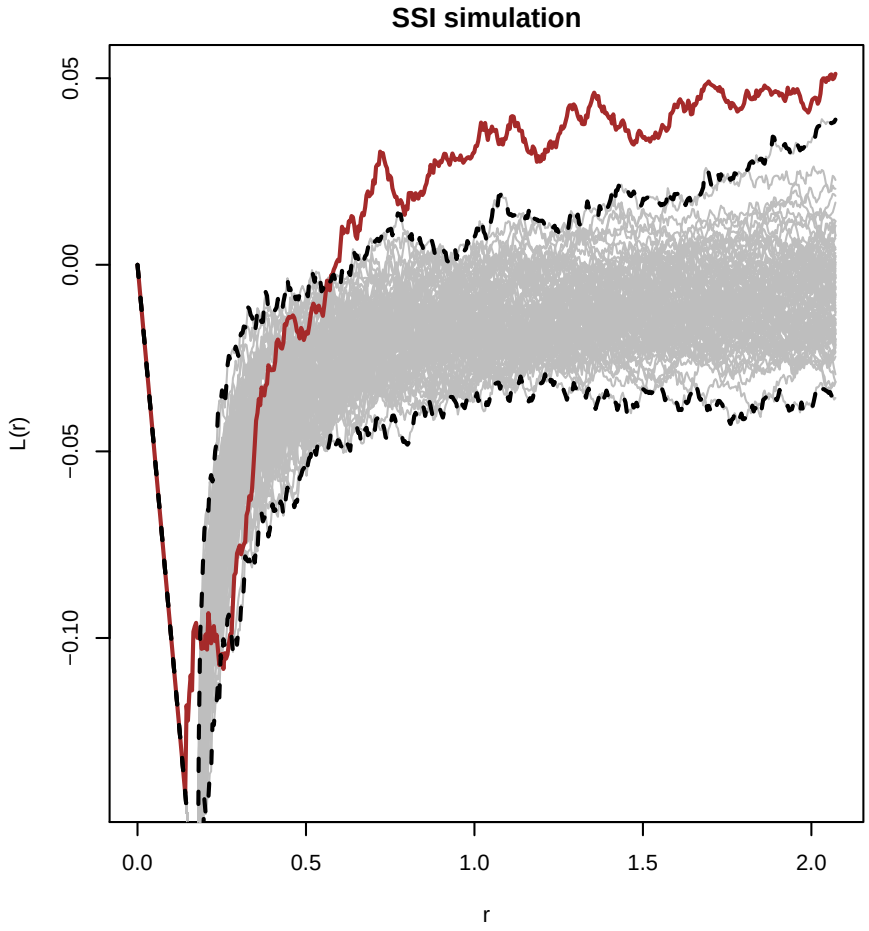
```
> ex <- expression(runifpoint(n,
+     win = rr))
> res <- envelope(drumlins_rr,
+     Kest, nsim = 99, simulate = ex,
+     verbose = FALSE, saveall = TRUE)
```



$\hat{K}$ with SSI simulation

From what we already know, drumlins represented as points appear to "inhibit" each other under a distance of about 200m, so running the $\hat{K}$ measure with an SSI process should show more of what is going on:

```
> ex <- expression(rSSI(0.18,
+     n, win = rr))
> res <- envelope(drumlins_rr,
+     Kest, nsim = 99, simulate = ex,
+     verbose = FALSE, saveall = TRUE)
```



Conclusions about drumlins (so far)

- Comparing the SSI with the CSR results indicates that we can not only reject the CSR process as being that driving drumlin point locations, but we have good grounds to suppose that a spatial inhibition process is operating
- the minimum drumlin "footprint" is such that drumlins cannot be closer to each other than a certain minimum distance
- At short range, the estimated $\hat{L}$ values are lower than the lower envelope bounds, but only less than 0.4 distance units — this corresponds to spatial inhibition
- As the $\hat{L}$ simulation values for the SSI process indicate, drumlins are not well represented by points, because they have area, even volume, and rarely "overlap" — but is there perhaps a trace of clustering at greater distances?

Conclusions about point patterns

- Point patterns are often observed: have we observed all the points?
- Are points a reasonable representation of the object we are interested in?
- What are the boundaries of our study area — does changing them make a difference?
- What kinds of processes can be generating our observed pattern — can we model them?
- Is our study area representative of the class of phenomena in more general terms?