

spatgraphs: Proximity Description of Spatial Point Patterns Using Graphs

June 18, 2010

Tuomas Rajala

1 Introduction

Great interest in spatial point pattern statistics is in the modeling of interactions of point cliques such as interacting pairs or triplets. The typical indication of interaction is given by Euclidian distance relation $\|\mathbf{x} - \mathbf{y}\| < R$ for some threshold parameter $R > 0$. This translates directly to a graph known as *geometric graph* (Penrose, 2003; Marchette, 2004). The information used by spatial point pattern summaries such as Ripley's K -function and empty space function (Illian et al., 2008) derives from the edge indication of such a graph. Alternative description of neighbourhood (e.g. *k*-nearest neighbours or *Delaunay* graph) might be of interest for example in heterogeneous or non-stationary settings, or when information such as individual point's size or type is thought to affect the neighbourhood.

The package **spatgraphs** is designed to work alongside such spatial point pattern analysis packages as **spatstat** (Baddeley and Turner, 2005). The goal is to aid in the exploration and adaptation of new interaction models by providing fast computation and visualisation of 12 node-location based graphs.

2 General usage

The accepted data is a pattern of distinct point locations in a bounded window. An object of class "ppp" from **spatstat** is recommended, but a list containing at least the coordinate vectors \mathbf{x} and \mathbf{y} suffices. The main function of the package is called **spatgraph**:

```
library(spatgraphs)
set.seed(12)
pp1<-list(x=runif(150), y=runif(150))
g<-spatgraph(pp=pp1, type="knn", par=2)
g$edges[[1]]
[1] 114 115
```

The object \mathbf{g} contains the connections in an element called **edges**. The connections are stored as a list of vectors, one vector for each point of the pattern. Each vector contains the indices of points to which the corresponding point is connected to. All information about proximity is in these vectors. So the neighbours of the i th point (x_i, y_i) are in $\mathbf{g}\$edges[[i]]$, and for example the neighbourhood

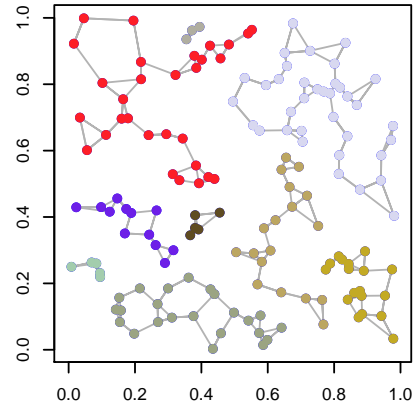


Figure 1: Poisson point pattern, 2-nearest neighbours graph and connected components.

sizes of all points can be computed with `'sapply(g$edges, length)'`.

The package also includes a function **spatcluster** which can be used to identify clusters (connected components) in a graph:

```
gc<-spatcluster(g)
gc$clusters[[1]]
[1] 115 114 10 46 1
```

Plotting methods are available:

```
# Figure 1
plot(g, pp=pp1, add=FALSE)
plot(gc, pp=pp1, add=TRUE)
```

There are three parameters for speeding up the calculation: **doDists**, **preprocessR** and **preGraph**. The first enables the pre-calculation of the pairwise distances to memory, second enables the pre-calculation of a geometric graph before the actual graph, and the third adopts the parameter as a pre-calculated graph from which to calculate the actual graph. The latter two are useful particularly for big datasets and complicated graphs as we can limit the search for connections to the R -close or otherwise chosen points. Another good application is when several spatial scales are of interest: The neighbourhoods are within each other and we can compute them in a descending order. For example, the following estimates the Ripley's K -function in 3D:

```
pp3d<-list(x=runif(150), y=runif(150), z=runif(150))
rvec<-seq(0,0.3, by=0.01)
g<-n<-NULL
for(r in rev(rvec)){
  g<-spatgraph(pp=pp3d, type="geometric", par=r,
               preGraph=g, toroidal=TRUE)
  n<-c(mean(sapply(g$edges, length)), n)
}
# Not included in this paper
plot(rvec, n/150, type="l", ylab="K")
lines(rvec, 4/3*pi*rvec^3, col=2, lty=2)
legend("topleft", c("K estimate", "Poisson"),
      lty=1:2, col=1:2)
```

The second line is the analytical Poisson process K -function, the flag `toroidal=TRUE` is described below.

3 Available graph structures

`spatgraph` can compute the following graphs:

| Name | Required parameter(s) |
|------------------------|-------------------------------|
| Geometric | threshold parameter R |
| Mark geometric | real valued marks |
| Mark crossing | real valued marks |
| Spheres of Influence | none |
| k -Nearest neighbour | count parameter k |
| Radial spanning tree | radiation origin (x_0, y_0) |
| Minimum spanning tree | none |
| Relative Neighbourhood | none |
| Gabriel graph | none |
| Class cover catch | types as marks |
| Delaunay triangulation | none |
| Signal-to-noise-ratio | See Dousse et al. (2005) |

For example the mark geometric graph, defined using relation $\|\mathbf{x} - \mathbf{y}\| < m(\mathbf{x})$ with $m(\mathbf{x})$ the mark of \mathbf{x} , allows each point to have it's on geometric radius.

In calculation the Euclidian distance $\|\mathbf{x} - \mathbf{y}\| = \sqrt{\sum_{i=1}^d (x_i - y_i)^2}$, $d = 2, 3$, is used and a toroidal version, which wraps the window on a torus such that left (bottom) and right (top) edge of the window have distance 0, is available for rectangular windows.

The package offer also a limited support for other graph structures. Adjacency matrices are handled with `sg2adj` and `adj2sg`. Small interface for package `igraph` is available:

```
library(igraph)
pp1<-list(x=runif(50), y=runif(50))
g<-spatgraph(pp1, "delaunay")
ig<-sg2igraph(g, pp=pp1)
lo<-cbind(get.vertex.attribute(ig, 'x'),
          get.vertex.attribute(ig, 'y'))
# not shown
tkplot(ig, layout=lo)
g2<-igraph2sg(ig)
```

Finally, one can export the connections to a DXF-file via command `'sg2dxf(x=g, pp=pp1, file="foo.dxf")'`. These functions are not directly related to spatial point pattern analysis but they allow data transfer between disciplines (network analysis) and programs (GIS, CAD).

4 Data-analysis example

I give here a short example how the package might be used in spatial point pattern analysis.

We wish to investigate the spatial mixing of nests of two species of ants, *Cataglyphis* and *Messor* (Harkness & Isham (1983)). We choose the 4-nearest neighbours graph to represent neighbourhood of individuals and calculate the mean fraction of alien nests in the neighbourhoods, a value known as the *mingling index*:

```
library(spatstat)
data(ants)

mingle<-function(pp, g)
{
  h<-numeric(pp$n)
  for(i in 1:pp$n){
    m<-pp$marks[g$edges[[i]]]
    h[i]<-sum(m!=pp$marks[i])/length(m)
  }
  h
}
g<-spatgraph(ants, type="knn", par=4)
h<-mingle(ants, g=g )
id<-ants$marks=="Messor"

# Figure 2 (top)
plot(g, pp=ants, add=FALSE,
     points.col=(1:2)[2-id],
     points.pch=c(19,17)[2-id])
legend("topleft", c("Messor", "Cataglyphis"),
      col=1:2, pch=c(19,17))

sapply(split(h, id),mean)
# 0.8103448 0.3125000
sapply(split(h, id),sd)
# 0.1965685 0.1949694
```

Top frame of Figure 2 depicts the data and the neighbourhoods. The mean fraction of alien neighbours is 0.31 for *Messor* and 0.81 for *Cataglyphis*, so a typical *Messor* nest has a smaller number of *Cataglyphis* nests in its vicinity than a typical *Cataglyphis* nest has *Messor* nests. But this is to be expected, as there are, in total, more *Messor* nests (68) than *Cataglyphis* nests (29).

Next we do a Monte Carlo test: We simulate 100 new patterns by permuting the marks on the original coordinates. This represent the hypothesis of no interaction, conditioned on the locations. For each such pattern we compute the mingling index, and get an estimate of the sample distributions:

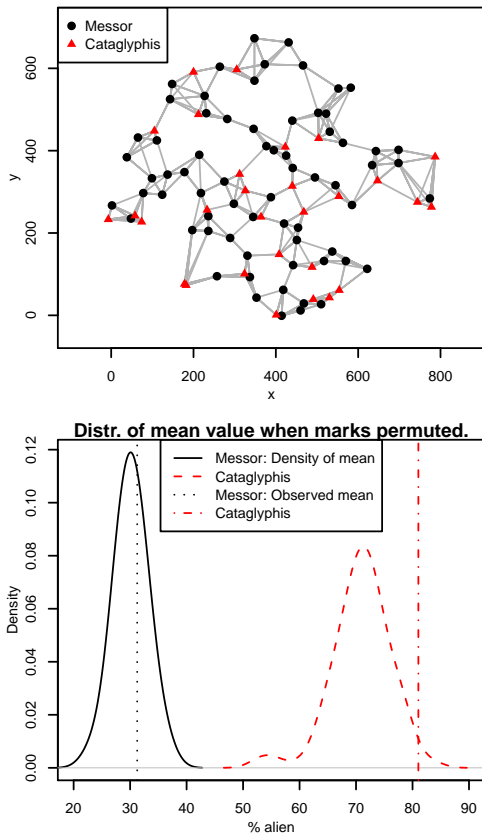


Figure 2: Top: Ants' nests and 4-nearest neighbours neighbourhoods. Bottom: % of alien neighbours in 100 simulations and in the data.

```

H<-NULL
set.seed(54321)
for(i in 1:100){
  pp<-setmarks(ants, sample(ants$marks,
    ants$n, replace=FALSE))

  h1<-mingle(pp, g)
  id1<-pp$marks=="Messor"
  H<-rbind(H, c(mean(h1[id1]), mean(h1[!id1])) )
}
means.messor<-H[,1]
means.cataglyphis<-H[,2]

#Figure 2 (bottom)
plot(density(100*means.messor, bw=2), xlim=c(20,90),
  xlab="% alien",
  main="Distr. of mean value when marks permuted.")
abline(v=mean(100*h[id]), col=1, lty=3)
lines(density(100*means.cataglyphis, bw=2),
  col=2, lty=2)
abline(v=mean(100*h[!id]), col=2, lty=4)
legend("top",
  c("Messor: Density of mean", "Cataglyphis",
    "Messor: Observed mean", "Cataglyphis") ,
  col=c(1,2,1,2), lty=c(1,2,3,4))

```

As can be seen from the bottom frame of Figure 2, The *Messor* nests in the data have similar neighbourhoods to those of the simulated patterns. This yields no evidence of

interaction. But the *Cataglyphis* nests in the data have a high portion of *Messor* nests around them when compared to the permuted patterns ($p \approx 0.01$), which implies that they are found near *Messor* nests too often to be there just by accident.

In this example the geometric neighbourhood would have described the scattering of points in relation to the window, something that we were not interested in. The topological k -nearest neighbours graph allowed us to focus better on the question of spatial mixing.

The neighbourhood is one of the important parts a researcher must decide upon when doing *any* kind of spatial analysis. The **spatgraphs** package provides an easy way to try different neighbourhoods (e.g. change `type="knn"` to `type="delaunay"` above) in spatial point pattern analysis, where problems of spatial heterogeneity and/or non-stationarity are often present and bias the simple geometric graph analysis.

References

- A. Baddeley and R. Turner. Spatstat: an R package for analyzing spatial point patterns. *Journal of Statistical Software*, 12(6):1–42, 2005. URL www.jstatsoft.org. ISSN 1548-7660.
- O. Dousse, F. Baccelli, and P. Thiran. Impact of interference on connectivity in ad hoc networks. *IEEE/ACM Transactions on Networking*, 13(2):425–436, 2005.
- R. Harkness and V. Isham. A bivariate spatial point pattern of ants' nests. *Applied Statistics* 32, 293–303.
- J. Illian, A. Penttinen, H. Stoyan, and D. Stoyan. *Statistical Analysis and Modelling of Spatial Point Patterns*. Wiley, 2008.
- D. Marchette. *Random Graphs for Statistical Pattern Recognition*. Wiley, 2004.
- M. Penrose. *Random Geometric Graphs*. Oxford university press, 2003.

Tuomas Rajala
 Department of Mathematics and Statistics
 University of Jyväskylä
 Finland
tuomas.a.rajala@jyu.fi

spatgraphs version 2.37