Intro to Spatial Statistics in R

David Diez

November 9th, 2009
Prerequisites

It is assumed that an attendant has...

- A strong understanding of basic probability theory.
- Taken a regression course.
- Familiarity with using R for regression and building very simple functions.
Outline

- Geostatistics
  - Regression modeling
  - Kriging
  - Correlogram (in passing)
- Point patterns
  - Poisson theory
  - Clustering
  - Tools to ID clustering
  - Inhibition
Geostatistical data includes observations at a collection of locations. The locations may or may not be random, and it is what is observed at these locations that is of particular interest.

Later we will discuss point pattern data, where the points themselves are the data (such as earthquake locations).
We will need the spatial and MASS packages:

```r
> library(spatial)
> library(MASS)
```

spatial is included in the initial R install, however, MASS is not. If loading MASS does not work, first install the package:

```r
> install.packages("MASS")
```
Data

Topographic heights within a 310 feet square. We follow the methods of *Modern Applied Statistics with S*. First, plot the data.

```r
> data(topo)
> plot(topo$x, topo$y, cex=(topo$z-500)/200)
```
We can fit polynomial models relatively easily. To fit a first degree polynomial, we use `surf.ls`.

```r
> plot(topo$x, topo$y, cex=(topo$z-500)/200)
> model1 <- surf.ls(1, topo)
```

Then we put the model in a form that can be easily plotted using `trmat` over the region of interest ($[0, 6.5] \times [0, 6.5]$):

```r
> mat1 <- trmat(model1, xl=0, xu=6.5, yl=0, yu=6.5, n=30)
> contour(mat1, levels=seq(600, 1000, 25), add=TRUE)
```
Linear model (the plot)
Linear model (summary)

We can examine the summary of the model:

```r
> summary(model1)
...
AIC: (df = 3) 378.5266
...
```

There are several aspects to look at in the summary but we will focus on the AIC as a measure of how well the model fits relative to its complexity:

\[ AIC = 2p - \log L(x) \]

where \( p \) is the number of parameters in the model. We seek a model with a small AIC (relatively).
2nd order model

We can fit a second order polynomial equally as easily:

```r
> model2 <- surf.ls(2, topo) # AIC: 357.5
> # etc...
```
Try it

While this second degree polynomial fits the data better than a linear model, maybe a higher degree fit will provide yet even more improvement.

Fit a polynomial of degrees 3 and 4, and plot the contours and identify the AIC of each.

You can plot 1 row and 2 columns of plots via the command (adjust for a different number of rows/columns):

> par(mfrow=c(1,2))

If you used the AIC as your guide for the best fitting model, which of these four models would you select?
What is Kriging?

In time series analysis, we sometimes use series autoregressive models in prediction:

\[ y_t = a_0 + a_1 y_{t-1} + \epsilon_t \]

Kriging does a spatial AR model of sorts, estimating the value at a location based on the other data points.
Kriging via `prmat`

First fit a GLS model and then use `prmat` instead of `trmat`:

```r
> topoGLS <- surf.gls(2, covmod=expcov, topo, d=0.7)
> krig <- prmat(topoGLS, 0, 6.5, 0, 6.5, 30)
> par(mfrow=c(1,2))
> contour(krig, levels=seq(600, 1000, 25))
> points(topo$x, topo$y, cex=(topo$z-500)/500)
```

We might also look at the estimate of the kriging errors:

```r
> SEs <- semat(topoGLS, 0, 6.5, 0, 6.5, 30)
> contour(SEs, levels=c(20,25))
> points(topo$x, topo$y, cex=(topo$z-500)/200)
```
Kriging result
We don’t have time for all of these, but this would be a good exercise for later. Load the library `geoR` and the data set `parana`. Change the format of the data set and plot the data:

```r
> Parana <- as.data.frame(cbind(parana$coords, parana$data))
> colnames(Parana) <- c('x','y','z')
> plot(Parana[,1:2], cex=(Parana[,3]-100)/300)
```

Fit a GLS model to the data and then perform Kriging. The following might be useful:

```r
> round(range(Parana$x))
[1] 150 769
> round(range(Parana$y))
[1] 70 462
```
Try it – solution

> pGLS <- surf.gls(2, expcov, Parana, d=0.7)
> krig <- prmat(pGLS, 150, 800, 50, 500, 30)

> par(mfrow=c(1,2), mar=c(2.5, 2.5, 1, 1))
> contour(krig)
> points(Parana$x, Parana$y, cex=(Parana$z-100)/300)

> SEs <- semat(pGLS, 150, 800, 50, 500, 30)
> contour(SEs, levels=seq(25, 35, 1))
> points(Parana$x, Parana$y, cex=(Parana$z-100)/300)
Try it – solution (cont)
Theory

Used in kriging is the covariogram, a measure of the distance between two observations:

\[ C(x, y) = c(d(x, y)) \]

Then \( c(r) \) is the covariation between observations a distance \( r \) apart. The correlogram is just the standardized covariogram.
Covariogram models

The exponential and Gaussian covariance models assume the covariance follows a parametric model:

\[ c(r) = \sigma^2 e^{-r/d} \quad \text{(exponential)} \]
\[ c(r) = \sigma^2 e^{-(r/d)^2} \quad \text{(Gaussian)} \]

where \( d \) is a parameter specified by the user.
topo correlogram of GLS residuals

```r
> correlogram(topoGLS, 25)
> r <- seq(0, 7, 0.1)
> lines(r, expcov(r, d=0.7))
> lines(r, gaucov(r,d=1,.3),
+ lty=2)
> legend('bottomleft', lty=2,
+ legend=c('exponential',
+ 'gaussian'))
```
Installing spatstat

We will use the library `spatstat` for spatial point patterns.

```r
> library(spatstat)
```

If you get an error, first install the package (and select a location to install from):

```r
> install.packages("spatstat")
```

This library has a wide range of point pattern functions, including simulation, summary, analysis, and modeling tools. (Among much more.) Help files for the package can be accessed via

```r
> help(spatstat)
```
What will we cover?

We will review

- basic Poisson process theory
- various processes and how to simulate them
- how to identify clustering and inhibition in point patterns
- very basic model fitting functions

The goal is to outline some basic tools for point patterns, which can be used to build up and/or test more complex custom code.
If $X$ follows a Poisson distribution with rate $\lambda$, then its PDF is

If $\lambda$ is large ($\lambda > 0$), then $X$ is approximately normally distributed with mean $\lambda$ and variance $\lambda$. 
Basics of Poisson processes

A common spatial model is a **Poisson Process** with intensity \( \lambda(x) \), i.e. the points arise at location \( x \) with “rate” \( \lambda(x) \).

- when \( \lambda(x) \) is large, a point at \( x \) is more likely
- a single location never has more than 1 point
- if \( \Lambda = \int_A \lambda(x) dx \) (the overall intensity in a region \( A \)), then the number of points in \( A \) follows a Poisson distribution with rate \( \Lambda \).
Example: Poisson process

To create a Poisson process with uniform intensity of 50 over $[0,1] \times [0,1]$,

```r
> pp0 <- rpoispp(50)
> plot(pp0)
```

To create the Poisson process with rate $50 \times (x^2 + y^2)$ over $[0,1] \times [0,1]$,

```r
> pp1 <- rpoispp(function(x,y)
+  { 50*(x^2+y^2) })
> plot(pp1)
```
The **ppp** class

The package `spatstat` uses a special class for points called **ppp**. The coordinates of the points in a **ppp** object can be accessed via

```r
> pp1$x
[1] 0.965859088 0.9874...
```

```r
> pp1$y
[1] 0.787572894 0.0075...
```

Objects of class **ppp** also maintain other components, including the space over which points are observed, the number of points in the pattern, and any “marks” associated with the pattern (for instance, earthquakes might be marked by their magnitude).
Pop quiz

Do any of the following patterns show clustering?
Pop quiz: solution

None are clustered processes. The points are all uniformly distributed over the space (generated from a uniform intensity, i.e. homogeneous, Poisson process).
Neyman-Scott

A Neyman-Scott process is basically a Poisson process with each “parent” point (red) replaced with $k$ uniformly distributed “children” points about it (black). Generate such a process via

```r
> par(mar=rep(0, 4))
> pp2 <- rNeymanScott(kappa=10, rmax=0.1,
+ function(x,y) runifdisc(5, 0.1, centre=c(x,y)))
> plot(pp2)
```
Matern clustering

Similar to Neyman-Scott except now the number of points is random. (There are more clustering models but we’ll stop here.)

```r
> par(mar=rep(0, 4))
> pp3 <- rMatClust(12, 0.1, 4)
> plot(pp3)
```
Chorley-Ribble Cancer Data

Spatial locations of cases of cancer of the larynx and cancer of the lung, and the location of a disused industrial incinerator. A marked point pattern.

```r
> data(chorley)
> plot(chorley)
```
Copper in Queensland, Australia

> data(copper)
> plot(copper$SouthPoints)
Try it

Use the function `rMatClust` to simulate a clustered process that is...

- clearly clustered, and name this pattern `clearlyClustered`.
- not clearly clustered but still might show some clustering, and name this pattern `littleClustering`.

When creating the second pattern, consider what properties would make it hard to distinguish it from a Poisson process. We will be using these patterns later on, so don't overwrite them!
It is useful to identify the type of process as either clustered or not. (Later we will discuss a process that is “anti-clustered”.) How might we do this?

A good first step would be to look at how close neighboring points are. A second step might be to look at the neighborhood intensity around points (as estimated by the data).
Nearest neighbor

For each point $p$, identify the distance $d_p$ of $p$ to its nearest neighbor. Then we can examine the distribution of these distances $d_p$ to see whether they follow what we would anticipate in a homogeneous (uniform) Poisson process.
If a process shows clustering, then the proportion of points with a small nearest neighbor distance will be larger (or is more likely to be larger) than would be anticipated under a Poisson process.

If the process is a Poisson process, then the proportion of points within radius $r$ should follow

$$G(r) = 1 - P(\text{no point within } r) = 1 - e^{-\lambda \pi r^2}$$
The function `Gest` may be used to estimate the empirical nearest neighbor function:

> plot(Gest(chorley))
Details

A small amount of information (a legend) is also printed in the command window.

- rs: the “reduced sample” or “border correction” estimator of $G(r)$
- km: the spatial Kaplan-Meier estimator of $G(r)$
- theo: the theoretical value of $G(r)$ for a stationary Poisson process of the same estimated intensity.
K function: neighborhood intensity

Of a similar nature, the K function estimates the intensity within $r$ of a location. If the process is homogeneous Poisson, then the K function $K(r)$ should closely resemble

$$\Lambda_{\text{within } r \text{ units}} = \lambda \pi r^2$$

$\lambda$ might be estimated via

$$\hat{\lambda} = \frac{\# \text{ of points in pattern}}{\text{area of space}}$$

The empirical K function can be compared to this theoretical function to identify clustering.
Poisson process

```r
> pp4 <- rpoispp(50)
> plot(pp4)
> plot(Kest(pp4))
```
Neyman-Scott

K function

pp5

Kest(pp5)
Inhomogeneous Poisson process

Even if we are looking at a Poisson process, there might be some apparent clustering.

This may be corrected with the weighted K function.
Inhomogeneous Poisson process

```r
> wts <- 50*(pp1$x^2+pp1$y^2)
> Kinhom(pp1, wts)
```

![Data points and Kinhom plot](image)
Matern I

One example of an inhibition process is a Matern I process. Produce a Poisson process with intensity \( \kappa \). If a point is within distance \( r \) of another point, delete it.

```r
> pp6 <- rMaternI(kappa=50, r=0.1)
> plot(pp6)
```
Try it

Make a Matern I process where consistently no points are produced, however, we require \( \kappa \) to be at least 1 and \( r \) to be no larger than 10% of the width of the space.
Nearest neighbor

```r
> data(cells)
> plot(cells)
> plot(Gest(cells))
```
Applying clustering techniques

**K function**

```r
> plot(cells)
> plot(Kest(cells))
```
Try it out

Examine the following data sets using both Gest and Kest:

- redwood and/or redwoodfull
- ants (this is a marked point process – for our purposes, ignore this marking)
- bramblecanes

Which show clustering? Which show inhibition?
Cluster model fitting

The package spatstat offers a variety of tools that can be used to estimate parameters of particular processes. We examine the utility of a couple of the functions.
Matern cluster process

```r
> pp7 <- rThomas(10, 0.03, 7)
> thomas.estK(pp7)
...  
  kappa sigma mu
9.85002358 0.03294598 5.68526558
```

David Diez
Intro to Spatial Statistics in R
UCLA Department of Statistics Statistical Consulting Center
Thomas process

```r
> pp7 <- rThomas(10, 0.03, 7)
> thomas.estK(pp8)
... kappa sigma mu
10.63830700 0.02599699 5.63999516
```

![pp8 plot](image.png)
Use the two processes you used to create patterns, clearlyClustered and littleClustered, and see how well the parameter fits are using matclust.estK.

How well does the model fit clearlyClustered? How well does it fit littleClustered?

Try making a new pattern that is somewhere in between these two patterns. Does the estimation work well?
Additional resources

The package geoR is a useful package that we did not examine. For information on this package, see

www.maths.lancs.ac.uk/~ribeiro/geoR.html

Book resource:

*Applied Spatial Data Analysis with R*
  by Roger S. Bivand,  
  Edzer J. Pebesma, and  
  Virgilio Gmez-Rubio
Please take our survey

Taking our survey to let us know what you think of mini-courses helps improve them for the future.

http://scc.stat.ucla.edu/survey