Geostatistics for spatial and space-time data

Ottar Bjørnstad

June 2, 2008

Introduction

Exploratory analysis of spatial data is of great interest in ecology of infectious disease for a number of reasons such as looking for space-time clustering, hot-spot detection, characterizing spatial waves. Spatial stats is also important in order to correct for the problem of spurious associations between incidence and environmental data because spatial autocorrelation can violate the assumption of independence.

Spatial statistics is a very field. In this tutorial we will focus on a subset of methods that are more (or less) commonly used. Many of these involves the notion of spatial autocorrelation in one form or another. While all the methods we will be discussing – such as Mantel tests, parametric and nonparametric correlation functions, local indicators of spatial association, etc – come in canned packages (we will be using the 'ncf'-package), it is useful to spend a bit of time on the underlying ideas.

Spatial autocorrelation

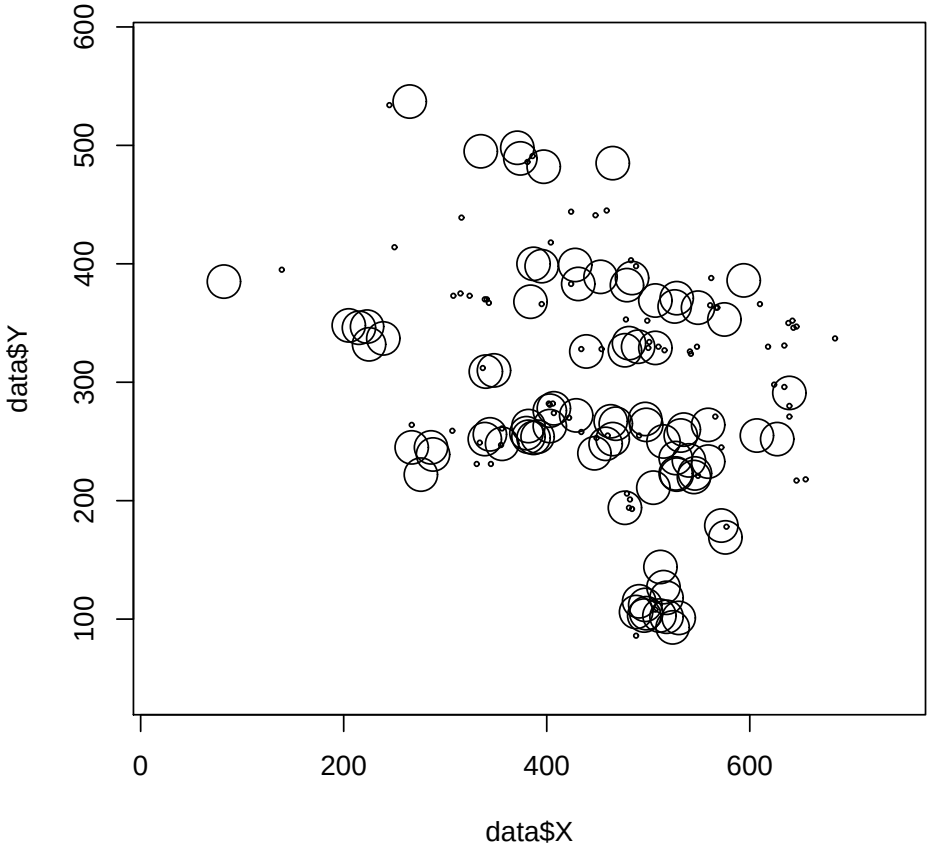
Pearson's product moment correlation between to random variables, Z_1 and Z_2, let's denote this by $\rho_1 2$ is defined as:

$$\frac{(Z_1 - \mu_1)}{\sigma_1} \frac{(Z_2 - \mu_2)}{\sigma_2}$$

where μ's are expectations and σ's are standard deviations. *auto*correlation has exactly the same definition and is used when the Z's are measurements of the same quantity (e.g. prevalence, incidence, presence/absence, etc) at different spatial locations (or different times).

To estimate this we need to know (or have an estimate of) the values of the μs and σs. In the case of single snapshot spatial data we use the *marginal* mean and *marginal* s.d. Let's explore using the filipendula rust data. The variable 'now' is presence/absence of the rust in 1994 and 'nxt' is presence/absence in 1995. 'X' and 'Y' are spatial coordinates.

```
> data = read.table("filipendula.txt", header = T)
> names(data)

 [1] "now"     "nxt"     "X"       "Y"       "P"       "D"       "PR"
 [8] "I"       "C"       "R"       "shore"   "expos"   "water"   "weather"
[15] "year"

> symbols(data$X, data$Y, circles = data$now, inches = 0.1)
```




```
> #marginal mean:
> mu=mean(data$now)
> #marginal sd:
> sig=sd(data$now)
```

The estimated 'autocorrelation matrix' among all 162 locations is then:

```
> zscale = (data$now - mu)/sig
> rho = outer(zscale, zscale)
```

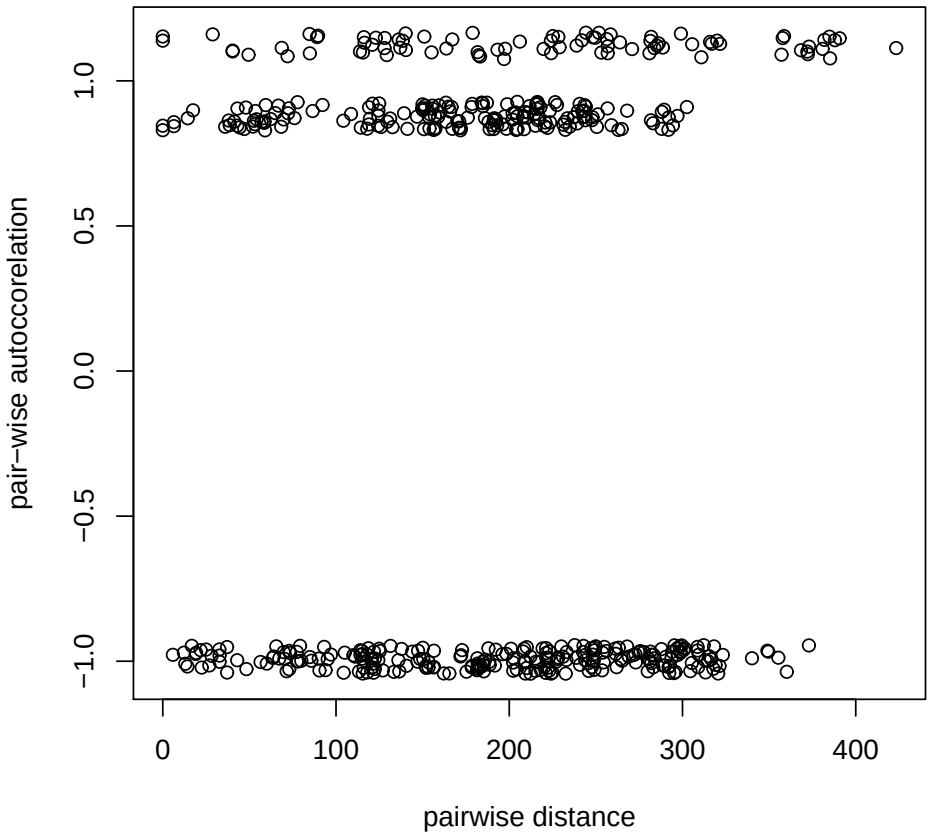
Note that since the data are binary presence/absence data, the 'individual pair-wise autocorrelation values' can only take one of three values – those arising from double-zero, double-ones and zero-one pairs. Note also that these individual values are not quite constrained to be between -1 and 1. This latter is not a worry though! Because the various different geostatistical methods we will be discussing involves simple manipulations of this matrix. [1]. For several of the methods we also need some sort of spatial distance matrix. Most commonly used is the Euclidian distance, but lots of other measures could be possible. The Euclidean distance matrix among all 162 locations is:

```
> dst = as.matrix(dist(data[, c(3, 4)]))
```

To understand the different geostatistical methods we will consider the plot of the first 500 pairs as a function of their spatial distance. (Plotting all the 26,000+ pairs would clutter up the screen). I have jittered the data to make the plot a little less cluttered.

[1] Note also that the geostats methods usually use the MLE estimator of sd rather than the BLUE estimator: i.e the denominator is n rather than $n-1$. This is of no importance in this introductory exposé.

```
> plot(jitter(as.vector(rho[1:500])) ~ as.vector(dst[1:500]),
+     ylab = "pair-wise autoccorelation", xlab = "pairwise distance")
```



NOW we are ready to conceptually understand the different geostatistical methods (We gloss over lots of bells and whisles here, of course):

- **Mantel test:** An overall test for whether there is any significant relationship between the two matrices. This is essentially a test of the linear regression of the data plotted in the figure.
- **Correlogram:** The most classic tool of testing how autocorrelation depends on distance whithout assuming any particular function – hack the 'x-axis' into segments (given by specifying some increment) and calculate the average within each distance class.
- **Parametric correlation functions:** Assume the relationship follows some parametric relationship – such as Exponential, Gaussian or Sperical functions – and do the appropriate nonlinear regression.
- **Nonparametric correlation function:** Fit a 'nonparametric regression' (usually a smoothing spline or a kernel smoother) to the relationship. This also goes by the name of the 'spline correlogram'.
- **LISA:** Local indicators of spatial association. A test for 'hotspots'. Specify a neighbourhood, and for each location calculate the average autocorrelation with all the other locations that belongs to its neighborhood.

There are a bunch of other named methods that are variations of these. Several of which are extentions to when there is mutiple observations at each location. In which case it is natural(?) to estimate the 'auto-correlation matrix' using the regular correlation matrix. The 'non-centred correlogram' is this multivariate

extention of the correlogram. The 'time-lagged spatial cross-correlation function' has been used to study waves of spread (see below). Various directional variations which allows the spatial correlation function to vary by cardinal direction. [2].

Testing and confidence intervals

An important reason why specialized packages are needed for these methods – despite why they are conceptually very, very simple – is because while the n original data-points may (or may not be) statistically independent, the n^2 numbers in the autocorrelation matrix is obviously very NOT statistically independent and the interdependence is very intricate. None of the classical ways of testing for significane or generating confidence intervals are therefore applicable. Testing is therefore usually done using permutation tests under the null-hypothesis of no spatial patterns. The correlogram (or Mantel test, or ...) of the real data should look no different than that for a random re-allocation of observations to the spatial coordinates. Statistical significance is calculated by comparing the observed estimate to the distribution of estimates for, say, 1000 different randomized datasets. If the observed is more extreme than, say, 950 of the randmized we conlude that there is significant deviation from spatial randomness at a nominal 5%-level. For some of the methods it is possible to generate confidence intervals using bootstrapping (resampling with replacement).

Practical examples

All the above methods are implemented in the 'ncf'-package in R.

Mantel test

Using the Fillipendula as a case study.

```
> test = mantel.test(x = data$X, y = data$Y, z = data$now)

> test

$correlation
[1] -0.02246385

$p
[1] 0.01698302

$call
[1] "mantel.test(x = data$X, y = data$Y, z = data$now)"

attr(,"class")
[1] "Mantel"
```

We see that there is a significant negative association between autocorrelation and distance. This is a crude tool but it does reveal that locations near each other tend to be more similar in disease status but those separated by a greater distance.

If we already have distance/simmilarity matrices, the syntax is:

```
> test = mantel.test(M1 = dst, M2 = rho)

> test
```

[2]The semivariogram is similar to the correlogram except that instead of using the 'autocorrelation similarity' measure it uses the 'semivariance dissimilarity' measure

```
$correlation
[1] -0.02246385

$p
[1] 0.01598402

$call
[1] "mantel.test(M1 = dst, M2 = rho)"

attr(,"class")
[1] "Mantel"
```

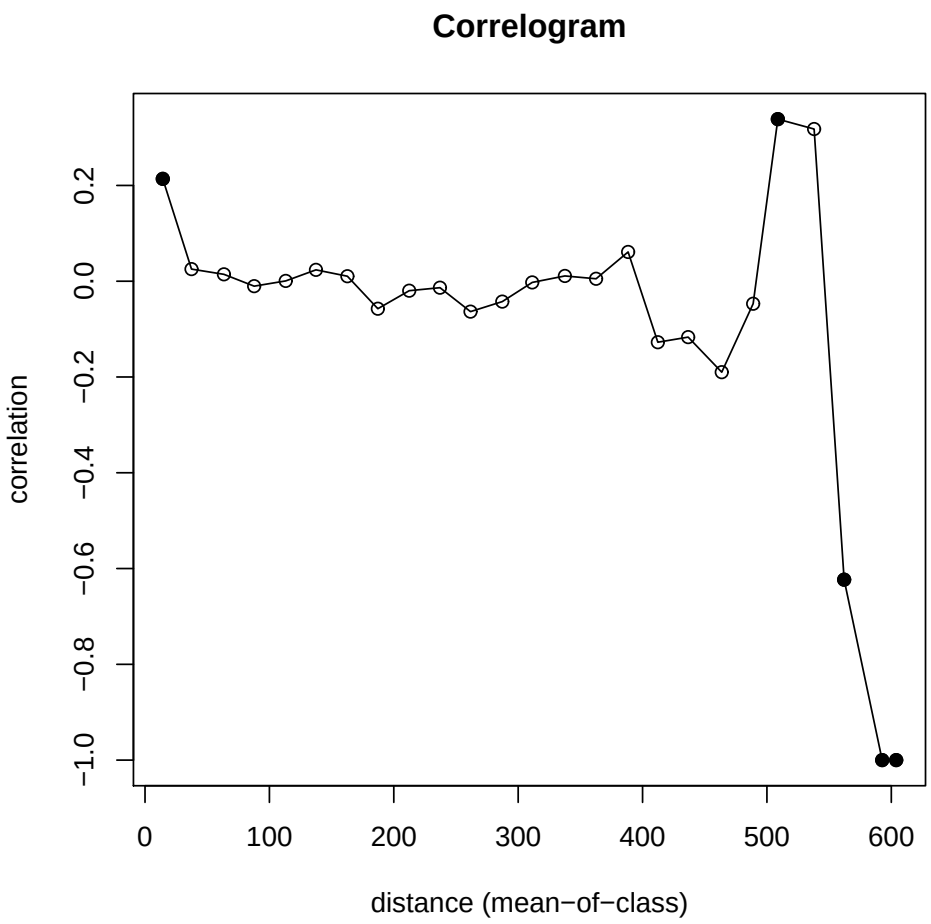
subsectionCorrelograms Now we first try the correlogram to look at how the autocorreltion is a function of distance

```
> test = correlog(x = data$X, y = data$Y, z = data$now, increment = 25)

> test

$n
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
 316  485  804  947 1249 1436 1204 1284 1152 1011  802  669  541  334  228
  16   17   18   19   20   21   22   23   24   25
 185  166   77   44   49   33   12   10    2    1

$mean.of.class
        1         2         3         4         5         6         7
 14.26854  37.35108  63.42978  87.72536 113.11611 137.42542 162.68037
        8         9        10        11        12        13        14
187.17729 212.40903 237.13161 261.67339 287.21247 311.30295 337.55785
       15        16        17        18        19        20        21
362.64639 388.38991 412.28721 436.67403 463.66415 488.91470 508.76005
       22        23        24        25
537.94131 562.06951 592.66480 603.91059

$correlation
            1             2             3             4             5
 0.2137199232  0.0252634103  0.0143681836 -0.0103476934  0.0006002354
            6             7             8             9            10
 0.0237433644  0.0104125579 -0.0574446059 -0.0197989979 -0.0135201886
           11            12            13            14            15
-0.0635170031 -0.0425789419 -0.0025203791  0.0110232412  0.0051294853
           16            17            18            19            20
 0.0611002680 -0.1274203300 -0.1167241571 -0.1899549349 -0.0470112157
           21            22            23            24            25
 0.3384332925  0.3177784578 -0.6232558140 -1.0000000000 -1.0000000000

$x.intercept
(Intercept)
   73.60151

$p
 [1] 0.001998002 0.226773227 0.276723277 0.401598402 0.418581419 0.113886114
 [7] 0.248751249 0.030969031 0.305694306 0.387612388 0.040959041 0.166833167
[13] 0.465534466 0.381618382 0.416583417 0.187812188 0.074925075 0.157842158
```

```
[19] 0.103896104 0.380619381 0.023976024 0.116883117 0.013986014 0.001998002
[25] 0.001998002

$call
[1] "correlog(x = data$X, y = data$Y, z = data$now, increment = 25)"

attr(,"class")
[1] "correlog"
```

The first distance class is significantly positiv. And the estimated distance to which the local positive distance decays to zero is 73.6 meters. Let's inspect:

```
> plot(test)
```



subsectionNonparametric spatial correlation functions We can get a bit finer resolution and confidence intervals for the underlying spatial correlation function through the spline correolgram. Note, here, that we only use 100 bootstrap resamples here too save some time – for a real analysis we wwould need 500 or a 1000 to get reasonable presicin for our

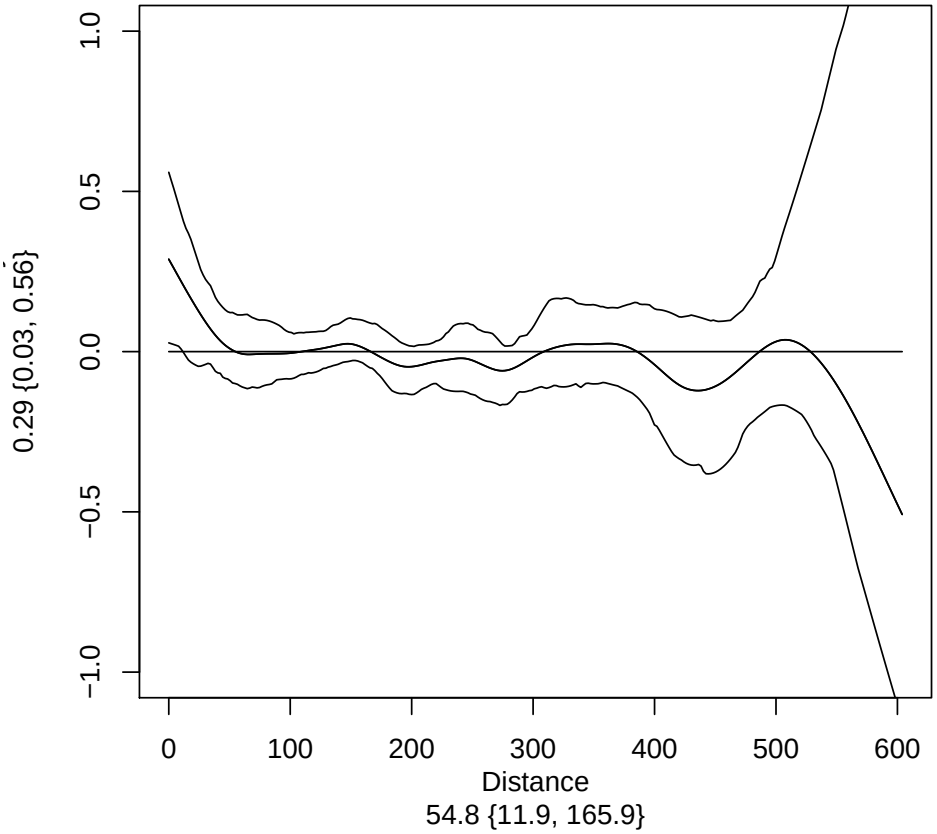
```
> test = spline.correlog(x = data$X, y = data$Y, z = data$now,
+     resamp = 100)

> summary(test)
```

```
$call
[1] "spline.correlog(x = data$X, y = data$Y, z = data$now, resamp = 100)"

$estimate
                x e         y
estimate 54.81374 0 0.2883401

$quantiles
               x         e           y
0     -102.98570  0.000000 -0.09851432
0.025   11.89752  0.000000  0.02733649
0.25    34.64078  0.000000  0.18430016
0.5     47.16534  0.000000  0.31279411
0.75    79.60309  3.331235  0.40419032
0.975  165.90515 16.419902  0.55954246
1      183.76266 23.430583  0.68741260
```

The spline correlogram returns a bunch of stuff – in fact all the summary statistics I thought might be of interest. These are:

- estimates: a vector of benchmark statistics
- x: is the lowest value at which the function is = 0. If correlation is initially negative, the distance calculated appears as a negative measure.
- e: is the lowest value at which the function is <= 1/e.
- y: is the extrapolated value at x=0.
- quantiles: A matrix summarizing the quantiles in the bootstrap distributions of the benchmark statistics.

```
> plot(test)
```



The plot shows the estimated correlation function with its bootstrap 95% confidence intervals. Since we have confidence intervals this allows us to compare different data sets.

Excercize: Try to estimate the spline correlogram for the 1995 data. Is there a difference between the 1994 and the 1995 patterns?

Two-species and space-time extensions

The 'Sncf'-function ('spatial nonparametric covariance function') is a function that allows for more complex (mysterious?). To explore some of these aspects, let's play with Aaron's CML model from yesterday. Recall yesterdays code:

```
> niter <- 520
> b0 <- 0.04
> b1 <- 0.8
> mu <- 0.02/26
> N <- 1000
> S.0 <- 100
> xlen <- 30
> ylen <- 30
> m <- 0.5
> local.dyn <- function(t, S, I, b0, b1, mu, N) {
+     beta <- b0 * (1 + b1 * cos(2 * pi * t/26))
```

```
+     I <- S * (1 - exp(-beta * I))
+     S <- (1 - mu) * S + mu * N - I
+     list(S = S, I = I)
+ }
> npatch <- xlen * ylen
> xy <- expand.grid(1:xlen, 1:ylen)
> d <- dist(xy)
> Psi <- matrix(0, npatch, npatch)
> Psi[(as.matrix(d) > 0) & (as.matrix(d) < 1.5)] <- 1
> D <- (m/8) * Psi + diag(1 - m, nrow = npatch)
> S <- array(0, dim = c(npatch, niter))
> I <- array(0, dim = c(npatch, niter))
> I[(xy[, 1] < 5) & (xy[, 2] < 5), 1] <- 10
> S[, 1] <- S.0 - I[, 1]
> for (t in 2:niter) {
+     tmp <- local.dyn(t, S = S[, t - 1], I = I[, t - 1], b0 = b0,
+         b1 = b1, mu = mu, N = N)
+     S[, t] <- D %*% tmp$S
+     I[, t] <- D %*% tmp$I
+     cat(t, " of ", niter, "\n")
+ }
> x <- xy[, 1]
> y <- xy[, 2]
```

We can use the Sncf to calculate the spatial correlation functions. However, we can also look at the spatial *cross-correlation* functions.

```
> fitI = Sncf(x = x, y = y, z = I[, 261:520], resamp = 0)
> fitS = Sncf(x = x, y = y, z = S[, 261:520], resamp = 0)
> fitSI = Sncf(x = x, y = y, z = S[, 261:520], w = I[, 261:520],
+     resamp = 0)
> par(mfrow = c(1, 3))
> plot(fitI)
> plot(fitS)
> plot(fitSI)
> summary(fitI)

$call
[1] "Sncf(x = x, y = y, z = I[, 261:520], resamp = 0)"

$Regional.synch
[1] 0.4566944

$Squantile
NULL

$estimates
            x  e        y    cbar
intercepts NA NA 1.047810 9.63875

$quantiles
NULL
```

The spline correlogram returns a bunch of stuff – in fact all the summary statistics I thought might be of interest. These are:

- Regional.synch: the regional mean (cross-)correlation.
- Squantile: the quantile distribution from the resampling for the regional correlation.
- estimates – a vector of benchmark statistics:
- x: is the lowest value at which the function is = 0. If correlation is initially negative, the distance calculated appears as a negative measure.
- e: is the lowest value at which the function is <= 1/e.
- y: is the extrapolated value at x=0.
- cbar: is the shortest distance at which function is = regional mean correlation.
- quantilesA matrix summarizing the quantiles in the bootstrap (or null) distributions of the benchmark statistics.

One interesting additional application is so-called time-lagged spatial correlation functions. For example we can look at the spatio-temporal relationship between the infecteds and themselves 5 time-steps later. This may help quantify wave-like spread.

```
> fitI = Sncf(x = x, y = y, z = I[, 261:515], w = I[, 266:520],
+     resamp = 0)
> plot(fitI)
```