Week 8.

Stat 222, Spatial Statistics. Lecture MWF 9am, Math-Sci 5203.
Professor: Rick Paik Schoenberg, frederic@ucla.edu, www.stat.ucla.edu/∼frederic

DAY EIGHTEEN. Monday, 5/21/01.

1) K function.

The theoretical K-function is defined as
$K(h) = \frac{1}{h}E[\#$ of points within h of a given point$]$.
You can estimate this in several different ways. One example is:
$\hat{K}_4(h) = \frac{1}{\hat{\lambda}N} \sum_{i \neq j} \frac{1}{w(s_i,s_j)} 1_{\{|s_i - s_j| \leq h\}}$,

where $\hat{\lambda} = \frac{n}{|A|}$, the average number of points per unit area over your whole region of observation, A,

and $w(s_i, s_j)$ is the proportion of the circle with center s_i passing through s_j that is inside the region of observation, A.

Note that the sum in the formula above is a double-sum; i.e. you sum over all pairs of distinct points i and j, so actually each pair of points gets counted twice.

The estimate $\hat{K}_4(h)$ gives pairs of points more weight if they are near the boundary of A. (Because w for such a pair of points will be small.)

$\hat{K}_4$ is approximately unbiased for small h, if N is isotropic.

The naive estimate $\hat{K}_1(h)$ ignores boundary effects:

$\hat{K}_1(h) = \frac{1}{\hat{\lambda}N} \sum_{i \neq j} 1_{\{|s_i - s_j| \leq h\}}$. The problem with $\hat{K}_1$ is that one is counting how many points are near a sample point s, but if s is near the boundary, then there is not as much opportunity for another point to be near s.

One could alternatively just avoid summing over points that are near the boundary. For instance

$\hat{K}_3(h) = \frac{1}{\hat{\lambda}} \sum_{i \neq j} 1_{\{|s_i - s_j| \leq h, d_j > h\}} / \sum_j 1_{d_j > h\}}$,

where d_j is the distance from point j to the nearest boundary of A.

Instead of summing over all (or nearly all) pairs of points, one could alternatively look at *random* collections of points in A and see how many points are near them, but that usually doesn't save much time and the resulting estimator would have higher variance (this is what Cressie is talking about on the bottom of p 640).

Note that for a stationary Poisson process, $K(h) = \pi h^2$, so $\sqrt{\frac{\hat{K}(h)}{\pi}}$ should be approximately h. To stabilize the variance, look instead at

$\hat{L}(h) = \sqrt{\frac{\hat{K}(h)}{\pi}} - h$,

which is an estimate of the theoretical L-function

$L(h) = \sqrt{\frac{K(h)}{\pi}} - h$.

So $\hat{L}(h)$ is centered around zero for stationary Poisson processes. If the point process

is clustered, one expects $\hat{L}(h) > 0$, and if $\hat{L}(h) < 0$, that suggests repulsive (inhibitive) behavior. See the example on top of p617 of Cressie.

Note that the edge correction in estimating the K-function matters a lot. See the bottom of p617: the estimated K-function that becomes negative is $\hat{K}_1$, with no edge correction.

One typically shows the estimated L-function along with confidence bounds, so that you can tell if the clustering or inhibitive behavior is statistically significant. See p617 for example.

How can one construct these bounds? The key thing is that, in making these bounds, you *assume* that the process is stationary Poisson, and see what the typical range of the estimated L-function is in that case. For certain observation regions A, such as circle or rectangle, one can apply analytic formulas to get the confidence bounds (see p642). For instance, if A is a circle of radius a, then under the assumption that the process is stationary Poisson,

$\mathrm{Var}(\hat{K}_4(h)) \approx \frac{2}{\lambda^2}(\frac{h}{a})^2[1 + .61(\frac{h}{a}) + .083\frac{\lambda h^3}{a}]$.

Instead, a procedure I recommend (because it works for more general regions A, and because these analytic formulas are only approximate anyway) is to obtain confidence bounds by simulation. For instance, one could simulate a stationary Poisson process on A 1000 times, and for each simulation, estimate the L-function. Then for each h, you have 1000 values of $\hat{L}(h)$; you can order these and take the 25th and 975th: these will give you the 95%-confidence bounds for $\hat{L}(h)$.

Usually, instead of simulating 1000 Poisson processes on A, you contrain your simulated processes to all have exactly n points, the same as your observed point process.

DAY NINETEEN. Wednesday, 5/23/01.

1) Projects.

a) There is still plenty of time for the projects. Don't worry.

b) I encourage you to WORK TOGETHER in figuring out how to do things in R.

c) R is easy to use and I will help show you how to do point process analysis in R. I thus recommend you use R for your projects. However, if you want to use a different program you may, but then I can't help you.

d) If you do not have an account in the stat dept, please email or see me so I can set up a way that you can use R. There are various options for this: I can give you an account on our statistics UNIX server and you can remotely log in to it, or I can give you an account to use on the mac clusters in the stat computing lab. Both of these already have R installed. Or, I can direct you to information on installing R on your own computer.

2) Grand canyon effect with k-functions.

If you estimate the K-function for a point process that has some trend (the rate at which points are accumulating is increasing or decreasing as you look in some direction within your observation region A), then the estimated K-function will generally indicate clustering even though there really may be no attractive effect. That is, the clustering may be just due to the trend. In order to see the interesting 2nd-order properties like attraction and repulsion between the points, one needs to remove the trend first. Analogously, one's initial view of the Grand Canyon is typically dominated by the main overwhelming feature which is the canyon's incredible size. The Colorado River at the bottom looks like a trickle. You need to neutralize this main feature in order to see more subtle features such as Lava Falls, a large waterfall within that portion of the Colorado River.

How can one *remove the trend* for a point process? Typically in statistics, the way to remove trend is to fit a model consisting of JUST a trend term, and then look at the residuals. In order to do that here, we have to know how to fit a model, and we need to discuss what residuals are for point processes. Also, in order to get confidence bounds for the resulting plot of the L-function on the residuals, we have to know how to simulate a Poisson process.

3) Parametric estimation.

Modeling a point process usually amounts to modeling the conditional intensity, λ. For a Poisson process, one could just write virtually any function to be the intensity, and this specifies the process exactly. There are just a few constraints: the conditional intensity must be non-negative everywhere, and also it must be left-continuous.

Given a model for the conditional intensity (which in the Poisson case is equivalent to simply the intensity λ_1) of a point process, one usually fits the model by choosing the parameters θ that maximize the likelihood of the observations. This is equivalent to maximizing the log-likelihood. The log-likelihood function $LL(\theta)$ is given by:

$$LL(\theta) = \int_A \log(\lambda) dN - \int_A \lambda(s) ds$$
$$= \sum_{i=1}^{n} \log[\lambda(s_i)] - \int_A \lambda(s) ds,$$

where the sum is over all observed points s_i.

Under rather general conditions, the resulting maximum likelihood estimate (MLE) is consistent, asymptotically normal, and efficient, as the observation region $A \uparrow \mathbf{R}^2$.

However, there are exceptions. For example, if $\lambda(s) = \alpha + \beta X + \gamma Y$, then if β or γ is negative the asymptotics don't make sense, since λ will eventually be negative. So instead sometimes people model

$\lambda(s) = exp\{\alpha + \beta X + \gamma Y\}$, i.e.

$\log\{\lambda(s)\} = \alpha + \beta X + \gamma Y$.

Then λ has to be non-negative everywhere. However, if β or γ is negative, the resulting estimates will not be consistent!

In order to remove trend, you will most likely want to fit a model such as those given above. I will leave it to you whether to fit the linear model to λ or to $\log\{\lambda\}$. The advantage of modeling λ as linear is that the parameters are easily interpretable and generally meaningful. The advantage of $\log\{\lambda\}$ is that it guarantees that the model is admissible as the observation size extends to the whole plane, and so the asymptotic results about the MLE can be applied.

4) Simulation.

a) Stationary Poisson. There are at least two ways to simulate a stationary Poisson process.

(i) First generate the random variable n, which will be the number of points in A. Generate a Poisson random variable with mean $\lambda|A|$ and let this be n. Then distribute the n coordinates *uniformly* throughout the region A. For a rectangular region A, this can be done just by drawing n iid uniform random variables and letting these be the x-coordinates of the points, and then doing the same for the y-coordinates.

(ii) Alternatively, for a rectangular region $(0, a_1) \times (0, a_2)$, one may start by generating the x-coordinates in ascending order. Generate iid exponential random variables with mean λa_2. Call these $e_1, e_2, \ldots$. Let $x_1 = e_1, x_2 = e_1 + e_2, x_3 = e_1 + e_2 + e_3, \ldots$ etc. until you get an x-coordinate that is bigger than a_1, i.e. that is outside of the observation region. Then generate the y-coordinates by generating iid uniform random variables on $(0, a_2)$.

These two methods are equivalent. Either one is fine with me.

b) Inhomogeneous Poisson process.

(Inhomogeneous means non-stationary.)

Suppose you want to simulate an inhomogeneous Poisson process on A. Suppose also that you know some number B, where $\lambda(s) < B$ for all s. Simulate a stationary Poisson process with rate B, and then go through all the points, keeping point s_i independently with probability $\lambda(s_i)/B$. You can figure out whether to keep each point by drawing iid uniform random variables on $(0, 1)$, and keeping the point if the uniform number that you drew is less than $\lambda(s_i)/B$.

DAY TWENTY. Friday, 5/25/01.

1) No class Monday, 5/28/01. Memorial Day.

2) Terms. "CSR" = Complete Spatial Randomness = stationary Poisson process. Homogeneous = stationary; inhomogeneous = non-stationary.

3) Projects.

a) By Wednesday, 5/30/01, try to have entered your data in R and have made a plot of your points, and compared them to the map of actual points if appropriate. EMAIL ME (frederic@stat.ucla.edu) WITH YOUR QUESTIONS on starting to use R! I'll be around all weekend, so please do not hesitate to ask me any questions.

b) In your final written project, I want 3-4 pages of text, double-spaced. NO MORE THAN 4 pages of text are allowed. See description at course website (http://www.stat.ucla.edu/courses/ , select "tests"). **After** the conclusion, have figures on separate pages at the end. Can have as many figures as you'd like.

c) Put 2-4 figures on each page. The one exception is the raw data, which for some of you is a map downloaded from the internet or obtained in hard copy; that can stand alone on one page if you'd like.

d) Pay attention to the **writing** of the project. Do not keep working on the figures and computations until the very last minute, so that the final report is sloppily written. Focus on trying to give the reader a clear picture of your data, highlighting the important features of the dataset.

e) Do point process analysis; avoid quadrat count analysis (see below).

4) More on simulations.

a) For confidence bounds for L-functions, constrain the simulations to have n points each. (so just distrubute the n points uniformly on A.)

b) Simulating non-Poisson processes.

If you know some number B, where $\lambda(s) < B$ for all s, then simulate a stationary Poisson process with rate B, and then go through all the points, in the order of your conditioning, keeping point s_i with probability $\lambda(s_i)/B$. If for example the conditioning is in terms of everything to the *left* of the given point s_i, then move to the right, from point to point, that is go in terms of points of increasing x-coordinates.

5) Quadrat counts.

One can analyze pps by binning up the number of points in each little portion, or pixel, or quadrat, of the observation region. One thus ends up with ordinary spatial data, where for each pixel, you have the number of points in that pixel. Then you can analyze these pixels by conventional means.

Don't do this in your projects! This is throwing away all the information on the locations of the points within the pixels! Often such information is important.

6) Residuals.

For an inhomogeneous Poisson process of intensity $\lambda(s)$ in the region $A = [0, a] \times [0, b]$, Cressie (p656) suggests looking at

$u_i = \int_0^{x_i} \int_0^b \lambda(x, y) dy dx$

or

$v_i = \int_0^a \int_0^{y_i} \lambda(x, y) dy dx.$

If the model $\lambda(s)$ is correct, then the points u_i should be like a realization of a homogeneous Poisson process on the **line**, with rate 1 point per unit line segment. Similarly, the points v_i should be like a realization of a homogeneous Poisson process on the line with rate 1.

More generally (and preferably) can get *spatial* residuals, as follows:

Take each point (x_i, y_i), keep its x-coordinate as-is, and move its y-coordinate to $\int_0^{y_i} \lambda(x_i, y) dy$. This results in a *residual* point process. Again, if the model for $\lambda(s)$ is correct, then the residual process should look like a homogeneous Poisson process with rate 1 on the transformed region (which may be irregular, not rectangular). For any x, the height of the transformed region is $\int_0^b \lambda(x, y) dy$.

In the above description, the residuals are obtained by stretching or compressing the y-coordinates. One could alternatively stretch or compress the x-coordinates, and leave the y-coordinates fixed.

The goal of making residuals is to form a process that is Poisson if and only if the model for $\lambda(s)$ is correct. Then you can inspect the residuals to see if the residuals look Poisson. What is nice about the *spatial* residuals is that, if the residuals are *NOT* Poisson, e.g. if they're clustered or something, then that will show up very clearly by plotting the residuals.

An alternative way to form residuals is by thinning: Find the minimum value m of $\lambda(s_i)$ over all points s_i in the inhomogeneous point process. Then take each point s_i in the inhomogeneous point process and keep it with probability $m/\lambda(s_i)$. The resulting residual process will be stationary Poisson (if the model for $\lambda(s)$ is correct), though not necessarily with rate 1. This thinning method of obtaining residuals is not as good as the other methods, because you lose some information. However, one could repeat the thinning procedure over and over, each time getting a different set of residuals, which is kind of interesting....

7) Some initial R functions.

To start R, just type R from the Unix prompt.
To quit R, type

q()

The only two weird things about R are that every function has to have parentheses (so for example the "q" function you use to quit has to have parentheses, even though it doesn't take any arguments). This distinguishes the function "q" from q, which could be a variable name if you wanted. The second weird thing is that equals (=) has a very special meaning in R, so to let $x = 7$, you can't just type $x = 7$. You have to use the $< -$ symbols instead of the = sign. This is the less than sign, followed by the minus sign.

I recommend that you use a text editor, to save all your commands, and cut and paste them into R.

INPUT DATA

x <− c(752.20, 593.40, 101.40, 357.80, 996.80, 48.05, 355.90, 710.30, 592.80, 770.10, 442.90, 371.80, 35.16, 901.10, 637.80, 158.60, 765.90, 326.20, 500.60, 812.20, 441.10, 831.50, 304.40, 743.30, 716.80, 990.70, 609.70, 825.20, 681.70, 422.40)

y <− c(860.90, 447.00, 166.50, 170.90, 730.60, 180.30, 151.10, 277.30, 514.00, 502.20, 584.80, 370.90, 833.20, 880.20, 447.40, 100.40, 44.70, 311.90, 837.70, 949.30, 89.24, 405.00, 578.40, 649.50, 27.59, 699.00, 664.30, 582.40, 590.60, 774.90)

Or make a file with x and y values and load it using ppinit (more later.)

PLOT DATA

plot(x,y)

TRY THESE:

plot(x,y,pch="x")
plot(x,y,pch="x",cex=.7)
plot(x,y,pch="*")

MISCELLANEOUS

length(x)
y
y[7]
help(plot)
help(par)

SAVING A PLOT

postscript("rickplot1.ps")
plot(x,y)
dev.off()

VIEWING AND CONVERTING TO PDF

Quit, and in Unix do:
ghostview rickplot1.ps
ps2pdf rickplot1.ps