Subject: counting points falling inside a Voronoi cell
Posted by paulartcoelho on Fri, 01 May 2015 15:33:02 GMT
View Forum Message <> Reply to Message

hey there,
i'm looking for ideas on how to best solve this.

I have 2 sets of x, y data that I will, non-creatively, call 'data1' and 'data2'. Using data1 I have built Voronoi cells/polygons to represent the x,y space. Now I need to discover how many points from data2 fall in each of the Voronoi cells built from data1. In other words, I need to be able to assign each of the data1's Voronoi cell to all points in data2 falling inside that cell.

Firstly I was suggested to use INSIDE function, but that doesn't look good to me because it involves going through a double loop: for every x,y point in data2, test each of the voronoi polygons until I discover a match. That FOR+WHILE structure will be slow, specially given that data2 is much larger than the number of Voronoi polygons (I'm "FOR-looping" through the largest of the data arrays, besides I know IDL "hates" loops :)).

I'm wondering how I can build an algorithm differently... if I think about the two datasets as if they were 2D images, I could FOR-loop through the Voronoi cells (the smallest dataset) and use them as a mask. It would be something like:

1) build a 2D image from data2
2) FOR loop: for each voronoi polygon, make a 2D image where the inside of the loop polygon equals to 1, everything else equals to 0
3) multiply the "voronoi mask" in step 2) by the image built in step 1)
The non-zero results of the multiplication in step above are exactly the points in data2 that I'm looking for (in true, I only need the total number of points, so i don't even need to track back to the original x,y data2 values).
4) Write out the number of non-zero points, close the FOR loop.
Done! (done in only 1 FOR loop, looping through the smallest dataset)

Does that reasoning make sense?
How can I efficiently build an image from a) a set of x,y points and b) a polygon?
If I don't find a way to efficiently built the images, better to stick to the inefficient loop over the larger x,y array using INSIDE I guess... (or not?)

thanks,
Paula