Subject: Re: Astronomys` Sixth Neighbour Needs Help
Posted by google_forums on Mon, 11 Aug 2003 14:53:50 GMT
View Forum Message <> Reply to Message

A slight twist to the Astronomer's problem...

I am running a similar loop to Astronomer, the exception is that rather than always needing the distance to the sixth closest point, the point that I need depends on another variable. What I am using this for is to spread points out across a map so that they are evenly distributed across the map while still maintaining the "best" points. For example, if I have a list of cities with Latitude and Longitude and population and I want to mostly display the largest cities, but where there are clusters of large cities (I.E. around Chicago) that are near eachother, I would only want to display the biggest, while if there was a smaller city in the middle of nowhere, I would want it to show. To do this I take a score factor (in this case population) and from each point, I calculate the distance to the nearest point that has a higher score... I then either use this as a weight factor along with the score, or I use it alone depending on how much I want the score to come through vs. how evenly I want the points spread out. My code below works fine, but as in Astronomer's case it is very slow -- I typically need to run this code for around 12,000 points but in some cases for up to 40,000 cases -- the latter, I have yet to have the patience for... in any case, here's the code:

```
pro closestpoint, data

; data is a preexisting structure that has fields for score (the
ranking
; field), latitude, longitude, id, and output fields to hold the
closest
; point with a higher field, the distance to that point, and a
"combined
; score" which weights the distance and the score


distance=data.distance
closest=data.closest
tmp=distance
k=0
for i=0L,n_elements(data)-1 do begin
  if k eq 0 then print, 'counter..............................',i, '
   ',data[i].id

  for j=0L,n_elements(data)-1 do begin
  lon1=data[j].longitude
```

```
   lat1=data[j].latitude
   lon2=data[i].longitude
   lat2=data[i].latitude


     if ((data[i].longitude eq data[j].longitude) and
(data[i].latitude eq data[j].latitude) $
         and (data[i].id ne data[j].id) and (data[i].score le
data[j].score)) then begin
     distance[i]=0.01
     closest[i]=data[j].id
     print, 'Station Colocation Found: ' + data[i].id + ' ' +
data[j].id + '     distance: ' + string(distance[i])
     endif else begin
           if ((data[i].score le data[j].score) and (data[i].id ne
data[j].id)) then begin

           tstdistance=map_2points(lon1,lat1,lon2,lat2, /miles)
           ;if k eq 0 then print, 'calculated
distance....',i,data[i].score, j,
data[j].score,tstdistance,distance[i]

               if ((tstdistance lt distance[i]) or (distance[i] eq 0))
then begin
               distance[i]=tstdistance
               closest[i]=data[j].id
                    if (k eq 0) then print, 'distance to ' +
data[j].id + ' = ' + string(tstdistance) +' closest=' + closest[i]
               endif
           endif
     endelse



   endfor

data[i].distance=distance[i]


k=k+1
if k eq 500 then k=0
endfor
data.combined=(data.distance * mean(data.score) / mean(data.distance)
* 4)+data.score
data.closest=closest
END
```