Subject: Dirty resampling of irregularly spaced point data with GRID_INPUT
Posted by Leo on Wed, 06 Aug 2014 09:02:55 GMT

View Forum Message <> Reply to Message

Dear group,

I have a dataset that consists of a large number of point measurements (n=10^6) that were taken along regular grid lines (500m x 500m), with single measurements taken with a spacing of about 0.5m (varying) along these lines.
I want to resample the dataset for interpolation to let's say points that are 25m apart. For now, I abuse GRID_INPUT with a large epsilon to find a subset of points and then do the resampling myself by finding for each measurement the closest one in the subset:

```
;call grid_input to remove "duplicates"
GRID_INPUT, x, y, z, xr, yr, zr, EPSILON=eps

;generate distance table
n = n_elements(x) & nr=n_elements(xr)
d=sqrt( (rebin(transpose(x),nr,n,/SAMPLE)-rebin(xr,nr,n,/SAMPLE))^2 + $
        (rebin(transpose(y),nr,n,/SAMPLE)-rebin(yr,nr,n,/SAMPLE))^2 )

;find closest in the subset
tt=min(d, di, DIMENSION=1)
di=(ARRAY_INDICES(d, di))[0,*]

;step through subset and aggregate
hh=HISTOGRAM(di, REVERSE_INDICES=ri)
xr=FLTARR(nr) & yr=FLTARR(nr) & zr=FLTARR(nr)
for i=0, N_ELEMENTS(hh)-1 do begin
  ii=ri[ri[i]:ri[i+1]-1]
  xr[i]=median(x[ii])
  yr[i]=median(y[ii])
  zr[i]=mean(z[ii])
endfor
```

However, I can't find information about how GRID_INPUT actually chooses the subset and I'd like to control what happens there...

Do you have any suggestions for a different solution? I thought about QHULL or TRIANGULATE to find the neighbors of each measurement, but many points are colinear. Any ideas for a straightforward solution?

Thanks a lot,

Leo

Subject: Re: Dirty resampling of irregularly spaced point data with GRID_INPUT
Posted by Leo on Thu, 07 Aug 2014 12:58:16 GMT
View Forum Message <> Reply to Message

OK, not much interest in this... I came up with a solution that works for me. However, I'm not sure if this is the right way of doing that. And it might need some tweaking...
x and y are coordinates, footprint is the desired spatial bin size.

```
;generate distance table
n = n_elements(x)
d=sqrt( (rebin(transpose(x),n,n,/SAMPLE)-rebin(x,n,n,/SAMPLE))^2 + $
        (rebin(transpose(y),n,n,/SAMPLE)-rebin(y,n,n,/SAMPLE))^2 )

iclose=INTARR(n,n)
igroup=LONARR(n)-1
g=0L

hh=HISTOGRAM(d, REVERSE_INDICES=ri, BINSIZE=round(footprint), min=0, NBINS=2)
for i=0, N_ELEMENTS(hh)-1 do begin
  if i eq 0 then k=1 else k=0 ; 1 is for points within footprint
  ii=ri[ri[i]:ri[i+1]-1]
  iclose[ii]=k
endfor

repeat begin
  ; find remaining points
  hh=HISTOGRAM(igroup, REVERSE_INDICES=ri, BINSIZE=1, min=-1, NBINS=1)
  remaining=ri[ri[0]:ri[0+1]-1] ;points that are not grouped yet

  ;choose new startpoint
  nclose=total(iclose,1,/INTEGER)
  tt=min(nclose[remaining],damin) ;next start point is the one with least close neighbors

  ;now check to which points this one is close
  hh=HISTOGRAM(iclose[remaining[damin],*], REVERSE_INDICES=ri, BINSIZE=1, min=0,
NBINS=2)
  ii=remaining[ri[ri[1]:ri[1+1]-1]] ;index to close neighbors

  ;and check which of these neighbors has max number of close neighbors and small distane to
them
  tt=total(iclose[ii, *], 2) *  1d/(1+sqrt(VARIANCE(x[iclose[ii]])+VARIANCE(y[iclose[ii]])))
  tt=max(tt,damax)

  ;ii[damax] is new center point
  ;store indices for this bin
  hh=HISTOGRAM(iclose[ii[damax],*], REVERSE_INDICES=ri, BINSIZE=1, min=0, NBINS=2)
  ii=ri[ri[1]:ri[1+1]-1]
  igroup[ii]=g & g++
  ;remove indices from the iclose (symmetric) array
  iclose[ii,*]=0 & iclose[*,ii]=0
endrep until total(iclose) eq 0 ;stop when no close points are left

;do cleanup
;...
;...  ?

;make bins
hh=HISTOGRAM(igroup, REVERSE_INDICES=ri)
nr=N_ELEMENTS(hh)
xr=FLTARR(nr) & yr=FLTARR(nr) & zr=FLTARR(nr)
for i=0, N_ELEMENTS(hh)-1 do begin
  ii=ri[ri[i]:ri[i+1]-1]
  xr[i]=median(x[ii])
  yr[i]=median(y[ii])
  zr[i]=mean(z[ii])
endfor
```