Subject: Re: Duplicate lat/long points
Posted by mccreigh on Fri, 16 Jan 2009 19:55:29 GMT
View Forum Message <> Reply to Message

Helen-
I've written a routine to do something very similar to what you are describing.
My problem, however may be significantly different (or not, depending on if you are really giving the full details).
Basically, I identify unique station lons and lats. The *extremely* complicating factor is that the same stations may be reported in 2 completely different precisions. For example 49N10' would not equal 49N10'15". But are these different stations or not? If you dont have this tolerance problem, then, use hist_nd. If your data are regularly spaced, and your hist not tremendously sparse then use hist_nd. Otherwise you may need to consider what I've done. I tried to use hist_nd in vein for this problem, so i thought it would be worth sharing. My problem requires a tolerance factor, eg any two stations separated by more than 1 minute are distinct. This translates into a bin size in histogram or in hist_nd. Even if your tolerance is relatively low (like 1 minute), the distance between any two points makes the size of the implied grid huge, ie the sparsity of the histogram output is overwhelming and you'll probably get and error. That's because of how huge the earth is and the implied grid in the histogram is massive with virtually no actual data on it. You have to use long64 to even represent numbers if you want them to be non-floating. It gets ugly, the problem is not a very clean one. It's one more level of complication you may need to consider before trying hist_nd. So hist_nd might not be the way to go, depending on how mesy your problem is. I think a searching and matching routine like i've used is much more practical for this described case. I posted my two codes (from a quick scan, i couldnt see any other externals in the programs, but there probably are, so just let me know on the side if i need to post those as well):

http://cires.colorado.edu/~mccreigh/idl/
qa_identify_unique_stations.pro
qa_find_data_precision.pro

the first calls the second in the case that the user does not supply a tolerance, in that case the second routine trys to find a default tolerance in the data.

But, since I love hist_nd so much, I thought I'd try to share a quick example like came up just the other day when I was sorting some dates.
But you really must/should study JD's histgram blub on David's site.
Afer that, to use hist_nd, you really only need read the header for hist_nd.

```
IDL> a=[3,3,3,4]      ; i guess you could think of these like dates -
day
IDL> b=[8,9,9,9]      ; month
IDL> c=[10,10,10,12]  ; year
IDL> in=[transpose(a),transpose(b),transpose(c)]  ;note the required
form of the input for hist_nd
IDL> hist=hist_nd(in,1,rev=ri)  ;binsize 1
IDL> p,hist
[*]
          1          0
          2          0

          0          0
          0          0

          0          0
          0          1

;; note the sparsity and structure - [2,2,3] is because 2 numbers span
variable1, 2 number span variable 2, 3 span the third
;; will illustrate the following reverse indices from the hist_nd
header (the trickiest part, make sure you read the hist primer)
;; ind=[i+nx*(j+ny*k)]
;;  ri[ri[ind]:ri[ind+1]-1]

IDL> dims=size(hist,/dim)
;you'd typically do this inside a "for ... do begin" loop and not
store the coeffs and it would look so much
;; nicer! but since I'm working from command line...
IDL> inds=intarr(dims[0],dims[1],dims[2])
IDL> for i=0,dims[0]-1 do for j=0,dims[1]-1 do for k=0,dims[2]-1 do
inds[i,j,k]=i+dims[0]*(j+dims[1]*k)
IDL> for i=0,dims[0]-1 do for j=0,dims[1]-1 do for k=0,dims[2]-1 do if
ri[inds[i,j,k]+1]-1-ri[inds[i,j,k]] ge 0 then $
IDL>        print,i,j,k,ri[ri[inds[i,j,k]]:ri[inds[i,j,k]+1]-1]
      0       0       0           0                    ; this says
that hist [i,j,k]=[0,0,0] has and entry in the input array at 0
      0       1       0           1           2          ; this says
that hist [i,j,k]=[0,1,0] has and entries in the input array at [1,2]
      1       1       2           3                    ; you
guessed it.
```

My only other routine in my code directory (variogram.pro) is a fancy wrapper on hist_nd. Hist is multithreaded, so this routine can run really well on a multiprocessor machine, one process will gladly take around 4 cpus on the machine I use. That's why programmers love histogram.

One last note, it may well be worth your while to ditch IDL for R,
unless you have serious memory requirements. R has way more libraries
and useful things available in general (from my experience). If you
dont have alot invested in IDL, I'd seriously consider R for a variety
of reasons (esp cause it's free). But IDL offers the satisfation of
writing your own programs in many cases!

Good luck, hope this is useful or at least reassuring if you've
already figured it out.

James