Subject: Hist_nd 3D +1 gridding / binning data
Posted by clement.feller@obspm. on Mon, 06 Mar 2017 22:42:29 GMT
View Forum Message <> Reply to Message

Hello everyone,

To go straight to the matter, I had a problem and found my solution. However I am writing to you for comments and advices.

Looking through the posts, I have read severals refering to the use of hist_nd or to that of reverse_indices, I also have found solutions using rebin/interpol for (longitude, latitude, temperature) problems, or refering to ncp and cic (from the astron library) or grid3.
But no express reference on the concurrent binning of 3 independants variables and 1 associated quantity. Do correct if I'm mistaken on that point, but in the meantime here's what I came up with.

From images, I have assembled a large table (4 columns of single-precision floats and about 160 millions lines) - 3 independants variables and 1 quantity - which I will later use to perform the inversion of a radiative transfer model through MPFIT.
Given hardware limitations, I sought to bin/resample/grid the data. Hence the following lines:

```
   density  = hist_nd([col1, col2, col3, col4], nbins=50, $
                  reverse_indices=ri) ;size(col1, /dimension) = [1,P]
   index = where(density ne 0, cts) ;finding non-empty bins

   newcol1 = fltarr(cts) ; a better way to allocate memory than density*0.
   newcol2 = newcol1
   newcol3 = newcol1
   newcol4 = newcol1

   for ijk=0L, (cts-1L) do begin
     init = ri[index[ijk]]
     stop = ri[index[ijk]+1L]-1L
     newcol1[ijk] = mean(col1[ri[init:stop]])
     newcol2[ijk] = mean(col2[ri[init:stop]])
     newcol3[ijk] = mean(col3[ri[init:stop]])
     newcol4[ijk] = mean(col4[ri[init:stop]])
   endfor
..... save data and move on to the next task
```

It takes about 15-20 secs to do the hist_nd task using 4 threads on a Intel Core i5-3230M CPU (3rd gen) @ 2.60GHz, which is pretty awesome.
But the averaging takes on a few hours, burning through all the cpu reserves.

Since my initial data are images, I binned them down to a 512x512 size (a fourfold reduction) and ended up with a table of 8.5 million lines instead.
In this case, hist_nd takes less than a second and the averaging takes about 15 minutes.

Do you have any advice, or have you ever tried to do that kind of task in a different way ?

I'll be looking forward to read your posts.
/C.

PS: For the python-enthousiasts out there which don't know it already, I found out that such a task can be achieved with the scipy.binned_statistics_dd method.

Disclaimer: What's pushing me to post and explicit this solution is that I was slow on the uptake from JD Smith's histogram tutorial and the documentation of hist_2d and hist_nd, that the reverse_indices vector is to be applied on *each* and *all* variable to get your data properly binned.
Yes, in the end, it's glaringly obvious but to quote JD, “one needs to learn to flex his/her histogram muscle.”
