Subject: Re: Finding the Top Two Most Common Coordinates in a Multi-Dimensional Array
Posted by Juggernaut on Wed, 30 Jul 2008 12:35:43 GMT
View Forum Message <> Reply to Message

On Jul 30, 7:54 am, Bennett <juggernau...@gmail.com> wrote:
> On Jul 29, 11:50 am, Jeremy Bailin <astroco...@gmail.com> wrote:
>
>
>
>> On Jul 29, 2:32 am, Brian Larsen <balar...@gmail.com> wrote:
>
>>> We do need some more information but this is just screaming for
>>> histogram. Have a read throughhttp://www.dfanning.com/tips/histogram_tutorial.html
>>> . Using histogram to see which x's are common you can step through
>>> the reverse_indices and see which y's are then common. There is
>>> probably a more graceful way however.
>
>>> Cheers,
>
>>> Brian
>
>>> ---------- ----------
>>> Brian Larsen
>>> Boston University
>>> Center for Space Physicshttp://people.bu.edu/balarsen/Home/IDL
>
>> In particular, if you're dealing with integers that don't span too big
>> a range, use HIST_2D and find the maximum element. If you've got
>> floats or a wide range, use UNIQ to turn each into an integer on a
>> small range first.
>
>> -Jeremy.
>
> I think if I were to be working with small datasets....ie not in the
> millions of points I would use something like this
>
> coords = [[10,1],[20,32],[5,7],[6,8],[20,32],[2,14],[20,32],[10,10],
> [3,1],[21,14]]
>
> counter = intarr(9)
>
> FOR i = 0, 8 DO BEGIN
> FOR j = 0, 8 DO BEGIN
>
> IF array_equal(coords[*,i],coords[*,j]) THEN counter[i]++
>
> ENDFOR

```
>  ENDFOR
>
>  ;- Histogram to find the max bins (no need to measure anything below 2
>  ;- because that would just be a single hit and if all of your pairs
>  ;- only occur once then who cares, right?
>  hist = histogram(counter, min=2, reverse_indices=ri)
>  maxHist = max(hist, mxpos)
>  IF maxHist EQ 1 THEN print, 'Each pair occurs no more than once'
>
>  ;- Use the reverse indices given by histogram to find out exactly
>  ;- where in your counter these maxes are occurring
>  array_index = (counter[ri[ri[1]:ri[2]-1]])[0]
>
>  ;- Find where counter is equal to the array index determined by
>  ;- reverse indices
>  max_index = where(counter EQ array_index)
>
>  ;- Voila with your max pair
>  print, coords[*,max_index[0]]
>
>  Which spits out....
>  20      32
>
>  This could be tweaked to find the top two or three or whatever as
>  well.
>  Hope this helps.
```

By the way....you shouldn't hard code these things as you can see that I've confused everyone by saying there are 9 pairs but there are actually 10. So replace those hard coded with n_elements(coords[0,*]) or size(coords, /dimensions) to get the correct loop numbers and array sizes. Still works though.
