Subject: Re: Finding the Top Two Most Common Coordinates in a Multi-Dimensional Array
Posted by Jeremy Bailin on Fri, 01 Aug 2008 11:02:09 GMT
View Forum Message <> Reply to Message

On Jul 31, 10:50 am, Bennett <juggernau...@gmail.com> wrote:
> On Jul 31, 7:37 am, Jeremy Bailin <astroco...@gmail.com> wrote:
>
>
>
>> 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.
>
>> My version of that would be:
>
>> min1=min(coords[0,*], max=max1)
>> min2=min(coords[1,*], max=max2)
>> arraymap = hist_2d(coords[0,*], coords[1,*], min1=min1, max1=max1,
>> bin1=1, min2=min2, max2=max2, bin2=1)
>> maxval = max(arraymap, maxelement)
>> print, array_indices([max1-min1+1,max2-min2+1], maxelement, /dimen)+
>> [min1,min2]
>
>> ...which avoids loops, and is more obvious to me.
>
>> -Jeremy.
>
> No loops is all and good...but if you put a decimal in coords like
> this
>

> coords = [[10.0,1.0],[20.0,32.3],[5,7],[6,8],[20.0,32.3],[2,14],
> [20.0,32.3],[10,10],[3,1],[21,14]]
>
> your code still spits out (20.0 32.0) where it should spit out (20.0
> 32.3)
> By the way the code I presented up there should have the following
> line replaced
> array_index = (counter[ri[ri[1]:ri[2]-1]])[0]
> with
> array_index = (counter[ri[ri[mxpos]:ri[mxpos+1]-1]])[0]

Like I said, if you have floats (or a very large range of integers),
you should map them into integers first using SORT and UNIQ...

```
coordsize = size(coords,/dimen)
coords0_sorted = coords[0,sort(coords[0,*])]
map0 = uniq(coords0_sorted)
nmap = n_elements(map0)
new_coords0 = lonarr(coordsize[1])
for i=0l,nmap-1 do new_coords0[where(coords[0,*] eq
coords0_sorted[map0[i]])]=i
```

...and the same for coords[1,*]. There's probably a more efficient way
of doing that, but you get the idea.

-Jeremy.
