Subject: Re: averaging over same index
Posted by JD Smith on Tue, 23 Aug 2005 19:53:05 GMT
View Forum Message <> Reply to Message

On Tue, 23 Aug 2005 11:41:46 -0500, Kenneth Bowman wrote:

```
> In article <1124814140.855435.78130@f14g2000cwb.googlegroups.com>,
>  andi.walther@web.de wrote:
>
>> Hello,
>>
>> I have an array V(alues) and an array S(ubscripts of a target array) and
>> I want to
>> extract the mean of all values with the same index in order to put them
>> into the new array.
>>
>> As a simple example:
>>   V =  [  3,  7,  99,  5,  2  , 10]
>>   S =  [  1,  3,  3,  2 , 0  ,1]
>>
>>   new vector should be  --> new = [ 2 , 6.5 ,  5, 53  ]
>>
>> slow way would be:  for n = 0 , max(S)-1 do new[n]=mean(v[where(S eq
>> n)])
>>
>>  a bit faster without WHERE in the loop:
>>
>>      VSorted = v[sort(S)]
>>     SSorted = S[sort(S)]
>>     uu  =  uniq(iSorted)
>>      for n = 0 , n_elements(uu)-1 do new[iSorted[uu[n]]] =
>> mean(vSorted[n:(n-1)>0])
>>
>> But is there a way to do what I want without resorting to a loop?  In my
>> real-world problem I have vectors with 500000 elements and the number of
>> the occurence of indices
>>  are quite irragular and can exceed 30 times.
>>
>>  Thanks Andi
>
> Use HISTOGRAM with reverse indices to get lists of each s.  Then average
> using array subscripting for each list.
>
> See JD's HISTOGRAM tutorial on David's web site.
>
> http://dfanning.com/tips/histogram_tutorial.html
```

And if you really have the need for speed, check out this thread where

we cover nearly all possible algorithms for this problem quite
exhaustively, many using HISTOGRAM (including the infamous
histogram-of-a-histogram method alluded to in the tutorial):

http://www.dfanning.com/code_tips/drizzling.html

I also dug out that old testing code and put it up here:

http://turtle.as.arizona.edu/idl/time_test_drizzle_alg.pro

This is a good point of embarkation after absorbing the HISTOGRAM
tutorial, going deeper into the potential uses of this function. It
also serves to illustrate a simple time testing setup.

Give it a run and see what's fastest for you (though do make sure you
understand the round-off issues with the sort cumulative method). For
mixed sparseness of indices in the same problem, from 1:20 to 30:1,
you'll probably have the best luck with one of the specialized
REVERSE_INDICES methods. Just plug `data' and `inds' in from your
problem. The sad and amusing thing is how much faster the one-line
literal loop approach in a compiled C DLM is (about 20x faster than
the fastest abstruse IDL method).

JD