Subject: Re: Faster way ?
Posted by btt on Mon, 27 Sep 2004 21:12:56 GMT
View Forum Message <> Reply to Message

Dick Jackson wrote:
> "Craig Markwardt" <craigmnet@REMOVEcow.physics.wisc.edu> wrote in message
> news:onr7onip35.fsf@cow.physics.wisc.edu...
>
>> rats@mail.geog.uvic.ca (Rafael Loos) writes:
>>
>>> Hi, I am trying to find the number of values that are within a range
>>> ...
>>> I have an Array that has 3 columns and 5 millions lines.
>>> Thats what I am doing ...
>>>
>>> number = WHERE((Array[1,*] GE Min) AND (Array[1,*] LE Max), geralX)
>>>
>>> I am storing the number inside the variable geralX ...
>>> It is taking 0.23 seconds ... but I want to know if there is a faster
>>> way to find that ...
>>
>> If you are doing this many times in a loop and ARRAY is unchanging, it
>> may be worth extracting ARRAY[1,*] into its own variable. That way,
>> you will save the time of extracting each iteration.
>>
>> If you just want the total number of elements that match your filter,
>> you can use total, as in:
>>
>> filter = (Array[1,*] GE Min) AND (Array[1,*] LE Max)
>> geralX = total(filter)
>
>
> Even with the two uses of Array[1,*], I got 30-40% time reduction with this:
>
> array1 = Array[1,*]
> number = WHERE((Array1 GE Min) AND (Array1 LE Max), geralX)
>
> ... and then splicing in your method gave a total of about 45% time
> reduction:
>
> array1 = Array[1,*]
> geralX = Total((Array1 GE Min) AND (Array1 LE Max))
>

Hello,

You know, if your data is composed of integers, then you canuse histogram pretty effectively (unless you expect to need a gazillion bins.) The steps below give

the following results...

Orginal Method (msec) 656.23403
Histogram Method (msec) 29.270887
Where Method (msec) 180.91989
Total Method (msec) 57.934046

As a bonus, you can use the REVERSE_INDICES keyword to get the locations of these values that fit your criteria. I suppose you could do this with floats, but then you have to fuss with where the bin locations start and how wide they are. But, I suppose if you do have floating decimal data but your cutoffs are fairly coarse, you could pull a fast one by bumping them up a could of orders of magnitude and then converting to integers.

For example, if your cutoffs are known to the second decimal place then maybe...

```
bumpedarray = LONG(array * 100)
bumpedMin = LONG(minVal * 100)
bumpedMax = LONG(maxVal * 100)
```

Now the histogram function might be handy (again, depends on what you data are like.)

Ben

```
***** BEGIN HERE
array = LONG(RANDOMN(seed, 3, 5000000L))

array1 = Array[1,*]
bottom = MIN(array1, max = TOP)
minVal = -2L
maxval = 2L


t0 = systime(/sec)
number = WHERE((Array[1,*] GE MinVal) AND (Array[1,*] LE MaxVal), geralX)
print, 'Orginal Method (msec)', 1000*(systime(/sec) - t0)


t0 = systime(/sec)
h = HISTOGRAM(array1,min = bottom, max = top, location = loc)
a = where(loc GE minval AND loc LE maxval, cnt)
if cnt GT 0 then geralX = TOTAL(H[A]) else geralX = 0
print, 'Histogram Method (msec)', 1000*(systime(/sec) - t0)

t0 = systime(/sec)
geralX = Total((Array1 GE MinVal) AND (Array1 LE MaxVal))
number = WHERE((Array1 GE Minval) AND (Array1 LE MaxVal), geralX)
```

```
print, 'Where Method (msec)', 1000*(systime(/sec) - t0)

t0 = systime(/sec)
geralX = Total((Array1 GE MinVal) AND (Array1 LE MaxVal))
print, 'Total Method (msec)', 1000*(systime(/sec) - t0)


*****FINI HERE
```
