Subject: Re: Entry comparison between two arrays
Posted by Chris[5] on Tue, 20 May 2008 19:38:54 GMT
View Forum Message <> Reply to Message

On May 20, 6:34 am, Jean H <jghas...@DELTHIS.ucalgary.ANDTHIS.ca>
wrote:
> crazywhiteboy...@gmail.com wrote:
>> Hello everyone,
>
>> I'm new to IDL (started using last week for a summer research
>> position), and have been working on a problem of comparing two
>> structures of arrays. The concept is that each of the structures
>> contain arrays of a year fraction, a day-of-year fraction, and other
>> data that I need compare. However, to do this comparison I need to
>> make sure both sets of data are corresponding to the same time (within
>> a range of error). My initial thought was to do this in a loop (only
>> have previous experience programming with Java and C doing small data
>> sets) that records the array indexes of corresponding pairs, and use
>> those indexes to build an another array that contains the two sets of
>> data from the two structures of arrays side by side in an array. This
>> idea works fine, but when I start trying to accomplish it on two
>> arrays of roughly 1 million entries each, it spikes up to taking my
>> whole summer to run. So what I'm looking for is how I might do this in
>> array based calculations in IDL. If anyone can give me an idea, or
>> point me in the direction of a good tool to use, I'd appreciate it
>> much. Thanks
>
> Hi,
>
> I guess you would have to use histograms.
> Read (several times) the famous histogram
tutorialhttp://www.dfanning.com/tips/histogram_tutorial.html and look, about
> mid page, at "Problem: Find the value intersection of two vectors,
> a.k.a. which values are present in both a and b?"
>
> Basically, you will want to do the same, but use a bin size equals to
> your error threshold (or twice, depending how you conceive it). Don't
> use "omin" in the first histogram, but min=min(data) - (1.0/2)*binSize,
> so each histogram bin will have the corresponding data + or - the error.
>
> You can play with the reverse index if you need the position, not the
> value, of the corresponding entries.
>
> Jean

Another approach: this task seems very much like the problem of
finding the closest match between two sets of objects (like star

positions). This has also been described on David Fanning's site (an excellent resource as you learn IDL, I might add) at http://www.dfanning.com/code_tips/slowloops.html.

The most straightforward IDL way to find the closest distance between two vectors of times (say t1 and t2) would probably be the following:

```
n1=n_elements(t1)
n2=n_elements(t2)
t1_array=rebin(t1,n1,n2)
t2_array=rebin(transpose(t2),n1,n2)

offset=abs(t1-t2)
```

This calculates the time offset between every pair of t1,t2 points. You could then find the smallest number in each offset column (the column is first index, which corresponds to the t1 value) - again see Fanning's webpage http://www.dfanning.com/misc_tips/max_row.html. You then have indices mapping every point in t1 to the closet point in t2, which you could compare to the time uncertainty to throw out bad points and continue.

Because IDL is well suited to do operations on entire arrays, calculating the offset of every pair of points is no sweat -- IF you have the memory. However, if each of your time list has a million or so elements, than the arrays have billions of points- it is very possible that IDL will not allow you to make such a big array (I think that's a function of both your system memory, and maybe some intrinsic IDL limit). Anyways, you could either break your arrays into smaller chunks and loop through each chunk using the above procedure, or check out the link to star matching.

Cheers,
chris
