Subject: Re: comparing and concatenating arrays...please help!!
Posted by JD Smith on Fri, 09 Jan 2004 18:12:20 GMT
View Forum Message <> Reply to Message

On Thu, 08 Jan 2004 03:27:57 -0700, Martin Doyle wrote:

> Hello all,
>
> I really hope someone out there can help me with this....I am tearing my
> hair out as my code is so slow!
>
> I have 2 files of data (hourly met data) with one file containing one
> set of parameters, and the other file containing another set of
> parameters. What I am trying to do, is to match the data based on the
> YY, MM, DD and HH values and then write BOTH sets of parameters to a
> seperate file. For example;
>
> file1:
> 1954 12 31 23  90  11  4  366  0.00
>
> file2:
> 1954 12 31 23  2.80  2.10  2.20  95.21
>
> intended result:
> 1954 12 31 23  90  11  4  366  0.00  2.80  2.10  2.20 95.21
>
> NOTE: Both files have no order to them, so a simple concatenation won't
> work
>
> I'm sure there must be a better way than this.

I predict this can be done in IDL in under 3 seconds. This is easy to convert into an "intersection of two arrays" problem: as Ben suggests, convert year, month, day, hour into a single long integer number (could be julian hours, could be hours since Jan 1, 1970, a long with all the data encoded in different bits, whatever). Read the entire file in at once (READCOL comes to mind) into separate vectors for each column, and perform this date conversion on the first 4. You now have two long integer vectors you'd like to match up, call them A and B. Read up on the various list intersection methods:

http://groups.google.com/groups?selm=38CBF8B6.5BF0AB50%40ast ro.cornell.edu

The last paragraph gives a nice synopsis of which to use: I'd expect either the SORT or HISTOGRAM methods will work. Stay away from the ARRAY method for such large data sizes. Your problem has one additional wrinkle: you don't just want the indices in A which exist

anywhere in B, you also want the matching indices in B. The HISTOGRAM method seems ideally suited to this, especially if your data come in regularly every hour, i.e. are not sparse (sometimes with an interval of an hour, sometimes two weeks), with a simple modification to capture the B indices:

```
function ind_int_HISTOGRAM, a, b, WHERE_B=whb
  minab = min(a, MAX=maxa) > min(b, MAX=maxb)
  maxab = maxa < maxb
  ha = histogram(a, MIN=minab, MAX=maxab, REVERSE_INDICES=reva)
  hb = histogram(b, MIN=minab, MAX=maxab, REVERSE_INDICES=revb)
  r = where((ha ne 0) and (hb ne 0), cnt)
  if cnt eq 0 then return, -1
  if arg_present(whb) then whb=revb[revb[r]]
  return,reva[reva[r]]
end
```

I tried this on two 250,000 long integer vectors which were about 1 in 4 sparse, and it took less that 1/2 second on my feeble laptop, which should nicely beat a Perl hash for data this regular (sparser or more random data is another story -- hashes are ideally suited for that):

```
IDL> b=long(randomu(sd,250000L) * 1000000L)
IDL> a=long(randomu(sd,250000L) * 1000000L)
IDL> t=systime(1) & wha=ind_int_HISTOGRAM(a,b,WHERE_B=whb) & print,systime(1)-t
      0.42473805
```

Also, if you want all the indices which are not in both A and B, look into the COMPLEMENT keyword to where, and use it in both instances above to return a WHERE_ONLY_A and WHERE_ONLY_B keyword in the same fashion.

JD

---