Subject: Re: Removing equal elements from an array
Posted by JD Smith on Wed, 16 Aug 2006 17:29:01 GMT
View Forum Message <> Reply to Message

On Wed, 16 Aug 2006 08:24:31 -0700, Julio wrote:

> Dear Maarten,
>
> I used your code to remove equal elements from an array. It worked fine
> for a small array. But I tested using a greater amount of points and
> some equal elements (pairs of coords) remains. There are 434 pairs...
> an example of them:
>
> 234.000 208.000
> 228.000 208.000
> 234.000 208.000
> 234.000 208.000
> 178.000 209.000
> ....
> 153.000 314.000
> 146.000 318.000
> 181.000 318.000
>
> The pair (234.000, 208.000) repeats 3 times, so 2 pairs should be
> removed. In the output array for these 434 input pairs I have:
>
> 234.000 208.000
> 228.000 208.000
> 234.000 208.000
> 178.000 209.000
> ... and so on
>
> We see the pair (234.000, 208.000) repeats 2 times! Do you have any
> idea about what is going on??

It's almost always a bad idea to rely on two floating numbers being precisely equal (as UNIQ does). See http://www.dfanning.com/math_tips/sky_is_falling.html. A better method is to test if they differ by less than some small number, epsilon.

Sadly, IDL's SORT isn't very flexible, since it only works on a single vector at a time, and you can't specify a generic sorting function. If your vectors cover a small range, and densely fill it, you can use HIST_2D or HIST_ND to bin them, taking the populated bin centers as your unique set, but this will become awkward for small bin sizes or very widely spaced data.

Maarten's code needs to add sort:

idx=uniq(I,sort(I))

but will still suffer from "almost equal" issues w.r.t round-off.

In this case, you know the (small) maximum range of your variables:
lon from 0-360, lat from -90 to 90 (I'm guessing; if not, it's easy to
generalize). It's therefore straightforward to create a single
lat_lon unsigned long long integer index which uniquely encodes the
latitude and longitude, and allows comparing coordinates to some
precision epsilon.

epsilon=1.e-7 ; difference in degrees for equality
lat_lon = ulong64((lat+90.)/epsilon) + ishft(ulong64(lon/epsilon),32)

As long as the "maximum value/epsilon" of either variable does not
overflow a 32 bit unsigned integer (e.g. 4294967295 or less), you'll
have a nice unique index. You may or may not actually want to use
epsilon=1.e-7; your data might be binned and truncated such that,
e.g. 1.e-3 or 1.e-4 is more appropriate. If comparing values to a
precision of 0.01 degrees is good enough, you can even squeeze lat_lon
into a normal 32 bit integer for some speedup (except on 64-bit
systems), since 360/.01 < 2.^16. In any case, this should work:

u=uniq(lat_lon,sort(lat_lon))
lat=lat[u] & lon=lon[u]

If you have three or more variables with similar range you want to
compare to some small epsilon, you should first understand MACHAR's
output, and especially that an absolute precision of, e.g., .01 is not
available at all floating point values, but in any case, you'll need
something else. Similarly if you have two floating-point numbers that
vary over a wide range.

JD
