Subject: Re: pointers and IF statement
Posted by peter.albert@gmx.de on Thu, 17 Nov 2005 07:30:10 GMT
View Forum Message <> Reply to Message

Hi,

I am not sure whether I get you right, but what you are looking for are those indices where both lists show the same value, right? I guess, with the line

WHERE(list1-list2(i) EQ min(list1-list2(i))

you try to find those indices where list1 is closest to list2[i]. If so, you should use

min(abs(list1-list2[i])) instead, as otherwise the minimum value is where the largest negative difference occurs, not the one closest to zero.

Apart from that, I'd strongly suggest to follow the suggestion to use square brackets [] for indexing and round brackets () for function calls. It makes the code better readable and helps avoiding confusion if you happen to name a variable after a function.

As for the error message, Ben's comment is perfectly right, generally WHERE returns an vector, so you have to check its first value against -1, which luckily does not throw an error if the rreturn value *is* -1, as you can index scalars with [0]. On the other hand, if you do something like

index = WHERE(this EQ that, n)

then "n" gives the number of matches, so you could continue with a line like

IF n gt 0 THEN do_the_rest

And now for your original problem. If you are dealing with integer values, I'd suggest using histogram instead of the for loop. Something like this:

Just a few example numbers:

list1 = [0,0,1,1,2,2,3,4,5,5,5]
list2 = [0,1,2,3,4]

We'll use histogram, so make sure to use the same min and max values in both calls:

```
min_value = min(list1) < min(list2)
max_value = max(list1) > max(list2)
h1 = histogram(list1, min = min_value, max = max_value, reverse = r1)
h2 = histogram(list2, min = min_value, max = max_value, reverse = r2)
```

Now, the key is the reverse index, which is just plain confusing when
you first look at it, but it just basically tells you the indices where
the list values do fall in each histogram bin.

If you want to see where list1 and list2 equal, say, 1, you can use

```
print, list1[r1[r1[1]:r1[2]-1]]
print, list2[r2[r2[1]:r2[2]-1]]
```

(just printing a lot of "1"s)

which means that
r1[r1[1]:r1[2]-1] gives you the indices where list1 equals 1, while
r2[r2[1]:r2[2]-1] gives the appropriate indices for list2.

Well, you are paying with weird looking code, but you'll gain a lot of
processing speed.

If this approach seems to work for you, then take the time to read JD
Smith's histogram tutorial at
http://www.dfanning.com/tips/histogram_tutorial.html

Cheers,

Peter
end