
Subject: Re: Strange WHERE issue?

Posted by [ben.bighair](#) on Fri, 27 May 2011 02:05:48 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hi,

On 5/26/11 8:52 PM, polystethylene wrote:

> Wonder if anybody can help me with this...

>

> Essentially I'm just varying a parameter (1st column) and testing chi
> squared (2nd column), and I want to find chi squared minimum + 1 in
> order find my 1 sigma error bars...

>

> I have the following code:

>

> c = WHERE(chisqarr_ph[1,*] EQ MIN(chisqarr_ph[1,*]))

>

>

> d = WHERE(chisqarr_ph[1,*] LE (chisqarr_ph[1,c]+1))

>

>

>

> I have a problem with the line d = WHERE(chisqarr_ph[1,*] LE
> (chisqarr_ph[1,c]+1)), which returns -1.

>

> What I don't understand is that when I print chisqarr_ph[1,c]+1, it
> correctly returns 1045.4516.

THis has to do with how IDL handles arrays of differing lengths. It turns out that your variable 'd' is an array (of length 1, so it looks like a scalar but it isn't). WHERE always returns a vector except when all of the values in the input argument are zero. (I know, it isn't consistent.) You want to make sure that you are using a scalar in the comparison as part of the comparison argument to WHERE. David Fanning has the scoop here...

http://www.idlcoyote.com/misc_tips/brokenwhere.html

It really isn't a WHERE issue, though. The meat of the issue is how IDL handles the mismatched array lengths.

Here, IDL 'knows' to scan the entire length of the first argument

```
IDL> print, [5,6,7,8] EQ 5
1 0 0 0
```

```
IDL> print, [5,6,7,8] EQ 7
```

0 0 1 0

Change the second argument to an array (of length 1) and things get trickier. In this case, IDL 'knows' to scan only the length of the shorter of the two arguments - comparing element-by-element.

```
IDL> print, [5,6,7,8] EQ [5]
1
IDL> print, [5,6,7,8] EQ [7]
0
```

It makes more sense when you add a second element to the second argument.

```
IDL> print, [5,6,7,8] EQ [5,7]
1 0
```

So, the 5s matched but the 6 and 7 did not. Bummer. This is why I often write variables that I think should be scalar as 'x[0]' - it looks silly but it prevents gotchas like this. Of course, there are times when one could exploit IDL's behavior - but I can't think of one right now.

Cheers,
Ben

```
> Here is the section of chisqarr_ph[*,*] where chisqarr_ph[1,*] is LE
> (chisqarr_ph[1,c]+1):
>
>
>    0.0030200630    1045.4305    6.5339409
>    0.0030400632    1045.3898    6.5336863
>    0.0030600633    1045.3513    6.5334455
>    0.0030800635    1045.3134    6.5332085
>    0.0031000637    1045.2760    6.5329750
>    0.0031200638    1045.2392    6.5327453
>    0.0031400640    1045.2011    6.5325071
>    0.0031600641    1045.1639    6.5322746
>    0.0031800643    1045.1274    6.5320465
>    0.0032000644    1045.0914    6.5318213
>    0.0032200646    1045.0560    6.5315999
```

>	0.0032400647	1045.0214	6.5313835
>	0.0032600649	1044.9917	6.5311983
>	0.0032800650	1044.9634	6.5310215
>	0.0033000652	1044.9358	6.5308485
>	0.0033200653	1044.9088	6.5306800
>	0.0033400655	1044.8827	6.5305167
>	0.0033600656	1044.8573	6.5303578
>	0.0033800658	1044.8333	6.5302083
>	0.0034000659	1044.8113	6.5300706
>	0.0034200661	1044.7896	6.5299349
>	0.0034400662	1044.7685	6.5298032
>	0.0034600664	1044.7481	6.5296754
>	0.0034800665	1044.7274	6.5295459
>	0.0035000667	1044.7072	6.5294197
>	0.0035200668	1044.6876	6.5292974
>	0.0035400670	1044.6687	6.5291792
>	0.0035600672	1044.6505	6.5290653
>	0.0035800673	1044.6329	6.5289559
>	0.0036000675	1044.6161	6.5288504
>	0.0036200676	1044.5997	6.5287478
>	0.0036400678	1044.5836	6.5286474
>	0.0036600679	1044.5683	6.5285520
>	0.0036800681	1044.5540	6.5284624
>	0.0037000682	1044.5403	6.5283766
>	0.0037200684	1044.5282	6.5283011
>	0.0037400685	1044.5186	6.5282411
>	0.0037600687	1044.5096	6.5281852
>	0.0037800688	1044.5013	6.5281333
>	0.0038000690	1044.4936	6.5280852
>	0.0038200691	1044.4866	6.5280414
>	0.0038400693	1044.4803	6.5280016
>	0.0038600694	1044.4746	6.5279665
>	0.0038800696	1044.4697	6.5279355
>	0.0039000697	1044.4653	6.5279080
>	0.0039200699	1044.4614	6.5278838
>	0.0039400700	1044.4581	6.5278632
>	0.0039600702	1044.4554	6.5278460
>	0.0039800704	1044.4532	6.5278327
>	0.0040000705	1044.4519	6.5278244
>	0.0040200707	1044.4516	6.5278223
>	0.0040400708	1044.4532	6.5278322
>	0.0040600710	1044.4574	6.5278584
>	0.0040800711	1044.4621	6.5278881
>	0.0041000713	1044.4674	6.5279212
>	0.0041200714	1044.4733	6.5279582
>	0.0041400716	1044.4787	6.5279918
>	0.0041600717	1044.4849	6.5280308
>	0.0041800719	1044.4919	6.5280741

>	0.0042000720	1044.4992	6.5281198
>	0.0042200722	1044.5071	6.5281691
>	0.0042400723	1044.5157	6.5282233
>	0.0042600725	1044.5272	6.5282952
>	0.0042800726	1044.5398	6.5283739
>	0.0043000728	1044.5530	6.5284564
>	0.0043200729	1044.5669	6.5285431
>	0.0043400731	1044.5815	6.5286344
>	0.0043600732	1044.5970	6.5287310
>	0.0043800734	1044.6138	6.5288366
>	0.0044000736	1044.6328	6.5289551
>	0.0044200737	1044.6521	6.5290758
>	0.0044400739	1044.6721	6.5292004
>	0.0044600740	1044.6926	6.5293290
>	0.0044800742	1044.7136	6.5294597
>	0.0045000743	1044.7351	6.5295941
>	0.0045200745	1044.7572	6.5297325
>	0.0045400746	1044.7800	6.5298750
>	0.0045600748	1044.8035	6.5300219
>	0.0045800749	1044.8277	6.5301733
>	0.0046000751	1044.8526	6.5303288
>	0.0046200752	1044.8779	6.5304871
>	0.0046400754	1044.9036	6.5306472
>	0.0046600755	1044.9299	6.5308120
>	0.0046800757	1044.9571	6.5309819
>	0.0047000758	1044.9849	6.5311559
>	0.0047200760	1045.0144	6.5313397
>	0.0047400761	1045.0466	6.5315415
>	0.0047600763	1045.0796	6.5317475
>	0.0047800764	1045.1132	6.5319574
>	0.0048000766	1045.1474	6.5321712

>

> Given that there are rows in the array for which the condition is

> true, can anybody see why WHERE is returning -1?

>

> It's probably me have a dim moment, but I can't for the life of me

> understand what's going on...

>

> Many thanks for any help !

>

> Stef