Subject: Re: Vector comparison.
Posted by JD Smith on Thu, 20 Nov 2003 19:38:51 GMT
View Forum Message <> Reply to Message

On Thu, 20 Nov 2003 10:35:48 -0700, Craig Markwardt wrote:

> David Fanning <david@dfanning.com> writes:
>> Timm Weitkamp writes:
>>
>>> Darnit! I was so proud of my answer... only to find that Chris Lee
>>> already proposed the same method (the "memory-hogging" one), albeit
>>> using more lines and this "if count eq 0 then ..." after WHERE, which
>>> always annoys me for some stupid reason. It can be avoided, which
>>> gives me an excuse to post my suggestion anyway:
>>>
>>> function Get_Match, a, b
>>> na = n_elements(a) & nb = n_elements(b) return,
>>> where(total((rebin(a,na,nb) - rebin(transpose(b), na, nb)) eq 0,
>>> 2))
>>> end
>>
>> Now there you go! That's a solution I can steal for my web page. :-)
>
> I think this is the same solution as Tom McGlynn. This is a classic
> solution by brute force *memory* hogging instead of compute hogging.
>
> If the lists are really large, then the arrays will be
> n_elements(a)*n_elements(b) in size, which can be pretty big. For
> example if each list had 10000 elements, then those 100 million element
> arrays would start to suck a large amount of virtual memory.

It's amazing how often this one comes up. I recall we hashed through this
ad naseum several times over the last couple of years.

In this post I recounted some earlier info on the problem, and gave three
comparable solutions, contrasting them by speed and features:

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

This includes the comment:

1. ARRAY is almost always slowest due to the large memory requirement.
Only for very small input vectors (10 elements or so), will ARRAY beat
SORT. It is, however, the only method which correctly identifies repeated
entries (not used in this test).

See this whole thread for more info. I've yet to find a non-loop,
non-array (i.e. memory hog) method of doing this with repeated indices.

JD