Subject: Re: Testing for NODATA presence in a dataset
Posted by David Fanning on Sat, 21 Dec 2002 22:20:19 GMT
View Forum Message <> Reply to Message

Jonathan Greenberg (greenberg@ucdavis.edu) writes:

> I'm having a problem testing for whether an entry in an array is NAN --
> doing something like:
>
> If (value EQ !VALUES.F_NAN) then begin
> print,'Not a number'
> Endif else begin
> print,'Is a number!'
> Endelse
>
> Will always return 'Is a number', even if I set:
> value = !VALUES.F_NAN
>
> What's going wrong with this?

The problem is that NAN is ... well, not a number.
Thus, you can't use it in expressions that
require a number. (Think of it as a mathematical
Catch-22, if you like.)

The proper way to write this code is like this:

```
 If Finite(value) EQ 0 then begin
    print,'Not a number'
 Endif else begin
    print,'Is a number!'
 Endelse
```

Cheers,

David

--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Testing for NODATA presence in a dataset
Posted by tam on Mon, 23 Dec 2002 02:33:44 GMT

David Fanning <david@dfanning.com> wrote in message
news:<MPG.186e97123b2778b2989a81@news.frii.com>...
> Jonathan Greenberg (greenberg@ucdavis.edu) writes:
>
>> I'm having a problem testing for whether an entry in an array is NAN --
>> doing something like:
>>
>> If (value EQ !VALUES.F_NAN) then begin
>> print,'Not a number'
>> Endif else begin
>> print,'Is a number!'
>> Endelse
>>
>> Will always return 'Is a number', even if I set:
>> value = !VALUES.F_NAN
>>
>> What's going wrong with this?
>
> The problem is that NAN is ... well, not a number.
> Thus, you can't use it in expressions that
> require a number. (Think of it as a mathematical
> Catch-22, if you like.)
>
> The proper way to write this code is like this:
>
> If Finite(value) EQ 0 then begin
> print,'Not a number'
> Endif else begin
> print,'Is a number!'
> Endelse

That doesn't distinguish NaN from the infinities.
The standard trick in any language for looking for NaN's is

```
if x ne x then begin
   print,'This is a NaN'
endif else ...
```

This can get optimized away if the compiler/interpreter
is poorly designed. Seemed to work for me in a quick
test though for IDL 5.2 on Linux. NaN's are not equal
to anything --- even themselves.

Regards,
Tom McGlynn

Subject: Re: Testing for NODATA presence in a dataset
Posted by Kenneth P. Bowman on Mon, 23 Dec 2002 14:42:26 GMT
View Forum Message <> Reply to Message

In article <f6352071.0212221833.611e00f5@posting.google.com>,
 tam@lheapop.gsfc.nasa.gov (Tom McGlynn) wrote:

> That doesn't distinguish NaN from the infinities.

The FINITE function has three keywords: NAN, INFINITY, and SIGN to
distinguish between NaNs, Infs, and to return the signs of arguments.

Ken Bowman

Subject: Re: Testing for NODATA presence in a dataset
Posted by thompson on Tue, 24 Dec 2002 15:11:32 GMT
View Forum Message <> Reply to Message

tam@lheapop.gsfc.nasa.gov (Tom McGlynn) writes:

> David Fanning <david@dfanning.com> wrote in message
news:<MPG.186e97123b2778b2989a81@news.frii.com>...
>> Jonathan Greenberg (greenberg@ucdavis.edu) writes:
>>
>>> I'm having a problem testing for whether an entry in an array is NAN --
>>> doing something like:
>>>
>>> If (value EQ !VALUES.F_NAN) then begin
>>> print,'Not a number'
>>> Endif else begin
>>> print,'Is a number!'
>>> Endelse
>>>
>>> Will always return 'Is a number', even if I set:
>>> value = !VALUES.F_NAN
>>>
>>> What's going wrong with this?
>>
>> The problem is that NAN is ... well, not a number.
>> Thus, you can't use it in expressions that
>> require a number. (Think of it as a mathematical
>> Catch-22, if you like.)
>>
>> The proper way to write this code is like this:
>>
>> If Finite(value) EQ 0 then begin
>> print,'Not a number'

>> Endif else begin
>> print,'Is a number!'
>> Endelse

> That doesn't distinguish NaN from the infinities.
> The standard trick in any language for looking for NaN's is

> if x ne x then begin
> print,'This is a NaN'
> endif else ...

> This can get optimized away if the compiler/interpreter
> is poorly designed. Seemed to work for me in a quick
> test though for IDL 5.2 on Linux. NaN's are not equal
> to anything --- even themselves.

> Regards,
> Tom McGlynn

I would like to echo Tom's response, and remind people that there is a whole family of values which are interpreted as NaN. The values returned by !VALUES.F_NAN and !VALUES.D_NAN are just the simplest forms. The "X NE X" syntax will catch them all, and has the additional bonus of working no matter whether the data is single precision, double precision, or even complex.

Bill Thompson

Subject: Re: Testing for NODATA presence in a dataset
Posted by David Fanning on Thu, 26 Dec 2002 16:25:15 GMT
View Forum Message <> Reply to Message

Tom McGlynn (tam@lheapop.gsfc.nasa.gov) writes and
Bill Thompson confirms:

> That doesn't distinguish NaN from the infinities.
> The standard trick in any language for looking for NaN's is
>
> if x ne x then begin
> print,'This is a NaN'
> endif else ...

Humm, well, consider this little test in IDL 5.5 or 5.6
for Windows:

```
IDL> a = [ 1.0, 2.0, !Values.F_NAN, 4.0, !Values.F_NAN ]
IDL> print, a
```

```
     1.00000      2.00000         NaN      4.00000         NaN
  IDL> print, a(1)
     2.00000
```

All well and good so far. Test the algorithm.

```
  IDL> if a(1) ne a(1) THEN print, 'NAN' ELSE print, 'Number'
     Number
```

Perfect. Working fine. Now text NAN.

```
  IDL> print, a(2)
      NaN
  IDL> if a(2) ne a(2) THEN print, 'NAN' ELSE print, 'Number'
     Number
     % Program caused arithmetic error: Floating illegal operand
```

Oh, oh. What's up with that? And a floating illegal operand to boot. :-(

How about the array in general?

```
  IDL> print, array ne array
     0   0   0   0   0
   % Program caused arithmetic error: Floating illegal operand
```

Humm. I presume you guys have a reason for thinking like you do. Any insights?

Cheers,

David

--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Testing for NODATA presence in a dataset
Posted by thompson on Thu, 26 Dec 2002 20:58:53 GMT
View Forum Message <> Reply to Message

Hmmm, you're right. This seems to be a Windows versus Unix thing. I just tried it myself on my workstation and my laptop, both in IDL 5.4, and the behavior under Windows was as you describe. I guess that means you have to use the FINITE() function after all.

Looking up the documentation, I realize that the keywords /NAN and /INFINITY were added in IDL version 5.2, so that FINITE(X,/NAN) is equivalent to X NE X, except that it also works under Windows.

In older versions of IDL, I guess you'd have to check explicitly for infinities to distinguish them from NaNs. Fortunately, there are only four different infinity values, as opposed to 9E15 different NaNs.

Sorry for giving wrong information,

William Thompson

David Fanning <david@dfanning.com> writes:

> Tom McGlynn (tam@lheapop.gsfc.nasa.gov) writes and
> Bill Thompson confirms:

>> That doesn't distinguish NaN from the infinities.
>> The standard trick in any language for looking for NaN's is
>>
>> if x ne x then begin
>> print,'This is a NaN'
>> endif else ...

> Humm, well, consider this little test in IDL 5.5 or 5.6
> for Windows:

> IDL> a = [1.0, 2.0, !Values.F_NAN, 4.0, !Values.F_NAN]
> IDL> print, a
> 1.00000 2.00000 NaN 4.00000 NaN
> IDL> print, a(1)
> 2.00000

> All well and good so far. Test the algorithm.

> IDL> if a(1) ne a(1) THEN print, 'NAN' ELSE print, 'Number'
> Number

> Perfect. Working fine. Now text NAN.

> IDL> print, a(2)
> NaN
> IDL> if a(2) ne a(2) THEN print, 'NAN' ELSE print, 'Number'
> Number
> % Program caused arithmetic error: Floating illegal operand

> Oh, oh. What's up with that? And a floating illegal operand to
> boot. :-(

> How about the array in general?

>   IDL> print, array ne array
>   0   0   0   0   0
>   % Program caused arithmetic error: Floating illegal operand

> Humm. I presume you guys have a reason for thinking
> like you do. Any insights?

> Cheers,

> David
> --
> David W. Fanning, Ph.D.
> Fanning Software Consulting, Inc.
> Phone: 970-221-0438, E-mail: david@dfanning.com
> Coyote's Guide to IDL Programming: http://www.dfanning.com/
> Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Testing for NODATA presence in a dataset
Posted by tam on Mon, 30 Dec 2002 14:52:21 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> Tom McGlynn (tam@lheapop.gsfc.nasa.gov) writes and
> Bill Thompson confirms:
>
>
>> That doesn't distinguish NaN from the infinities.
>> The standard trick in any language for looking for NaN's is
>>
>> if x ne x then begin
>>   print,'This is a NaN'
>> endif else ...
>
>
> Humm, well, consider this little test in IDL 5.5 or 5.6
> for Windows:
>
>   IDL> a = [1.0, 2.0, !Values.F_NAN, 4.0, !Values.F_NAN]
>   IDL> print, a
>   1.00000   2.00000   NaN   4.00000   NaN
>   IDL> print, a(1)
>   2.00000

>
> All well and good so far. Test the algorithm.
>
> IDL> if a(1) ne a(1) THEN print, 'NAN' ELSE print, 'Number'
> Number
>
> Perfect. Working fine. Now text NAN.
>
> IDL> print, a(2)
> NaN
> IDL> if a(2) ne a(2) THEN print, 'NAN' ELSE print, 'Number'
> Number
> % Program caused arithmetic error: Floating illegal operand
>
> Oh, oh. What's up with that? And a floating illegal operand to
> boot. :-(
>
> How about the array in general?
>
> IDL> print, array ne array
> 0 0 0 0 0
> % Program caused arithmetic error: Floating illegal operand
>
> Humm. I presume you guys have a reason for thinking
> like you do. Any insights?
>
> Cheers,
>
> David

Just to follow up on Bill's message.... I did warn in my first message that interpreters had been known to screw up this comparison, but I believe the behaviour you see above is clearly non-compliant with the IEEE 754 floating point standard. I almost never run IDL under Windows, but I'd call this a bug -- though I daresay RSI will call it a feature.

Using IDL 5.2 under Linux I have:

```
IDL> a=sqrt(-1)
%Program caused arithmetic error.  Floating illegal operand.
IDL> print, a
        -NaN
IDL> print a ne a
   1
IDL> z=[0,0,a,a,0]
IDL> print, z ne z
   0   0   1   1   0
```

I believe this to be 'correct' behavior but it appears that
it is not universally implemented this way within IDL. Of
course IDL has been implemented on non-IEEE machines (e.g., VAX)
and so completely consistent behavior may be impossible.

Let me add my apologies though for misleading anyone looking for
how to actually do something, rather than how they should be able
to do it.

Regards,
Tom McGlynn
