Subject: Re: BYTSCL and NAN keyword
Posted by Edd Edmondson on Tue, 02 Mar 2004 17:13:09 GMT
View Forum Message <> Reply to Message

Kenneth Bowman <k-bowman@null.tamu.edu> wrote:
> Has anyone else noticed this problem with BYTSCL and the NAN keyword?

> The BYTSCL function scales integers and floats into bytes between 0 and
> 255 (or the value set by the TOP keyword). If the NAN keyword is set,
> NANs in the input array are set to 0 in the output array. But 0 falls
> into the valid range for good values (0 to TOP)!

> Because it is not possible to set NANs to a value outside the valid
> range (greater than TOP), it is not possible to distinguish missing from
> valid data.

> As best I can tell, the only solution is to not use the NAN keyword and
> scale the valid data only by using WHERE to find all the valid values.
> Am I missing something obvious? (Quite possible, I admit.)

Yes, remember the NaN locations *before* you BYTSCL. Use
http://www.dfanning.com/tips/check_nan.html
to do this.

Then afterwards you can do something to the BYTSCLd array to put the NaNs
back in appropriately.

--
Edd

Subject: Re: BYTSCL and NAN keyword
Posted by David Fanning on Tue, 02 Mar 2004 17:18:14 GMT
View Forum Message <> Reply to Message

Kenneth Bowman writes:

> Has anyone else noticed this problem with BYTSCL and the NAN keyword?
>
> The BYTSCL function scales integers and floats into bytes between 0 and
> 255 (or the value set by the TOP keyword). If the NAN keyword is set,
> NANs in the input array are set to 0 in the output array. But 0 falls
> into the valid range for good values (0 to TOP)!
>
> Because it is not possible to set NANs to a value outside the valid
> range (greater than TOP), it is not possible to distinguish missing from
> valid data.
>

> As best I can tell, the only solution is to not use the NAN keyword and
> scale the valid data only by using WHERE to find all the valid values.
> Am I missing something obvious? (Quite possible, I admit.)

Well, NANs are definitely floats (that is, they have a float-type
bit pattern). So it seems reasonable to me that if you are
trying to stuff it into a byte (which can only have values between
0 and 255, that 0 is a good choice. (The only possible other
choice is 255, but that just turns your problem on its head.)

I think what I would do is locate the NANs and save their
indices. Then scale my data into 255 colors (0 to 254), leaving
color index 255 for the NAN color (whatever that is). This will
take a couple of steps, but that's what IDL programs are for. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: BYTSCL and NAN keyword
Posted by K. Bowman on Tue, 02 Mar 2004 17:26:19 GMT
View Forum Message <> Reply to Message

In article <MPG.1aae6fc4ef56f7829896c3@news.frii.com>,
David Fanning <david@dfanning.com> wrote:

> Well, NANs are definitely floats (that is, they have a float-type
> bit pattern). So it seems reasonable to me that if you are
> trying to stuff it into a byte (which can only have values between
> 0 and 255, that 0 is a good choice. (The only possible other
> choice is 255, but that just turns your problem on its head.)
>
> I think what I would do is locate the NANs and save their
> indices. Then scale my data into 255 colors (0 to 254), leaving
> color index 255 for the NAN color (whatever that is). This will
> take a couple of steps, but that's what IDL programs are for. :-)

Right, I understand all that. I'm just trying to understand what the
NAN keyword is good for.

If you could say NAN = 255 (or 19, or whatever), instead of just /NAN,
it would be of some use. That way the value that NANs become in the

output could be distinguished from valid data.

As it is, the NAN keyword is only of use if you are happy with NANs turning into valid data in the output (i.e., indistinguishable from small valid values).

I have to consider this an error in the way the handling of NANs was implemented in BYTSCL.

Ken

Subject: Re: BYTSCL and NAN keyword
Posted by Craig Markwardt on Tue, 02 Mar 2004 17:39:16 GMT
View Forum Message <> Reply to Message

Kenneth Bowman <k-bowman@null.tamu.edu> writes:

> Has anyone else noticed this problem with BYTSCL and the NAN keyword?
>
> The BYTSCL function scales integers and floats into bytes between 0 and
> 255 (or the value set by the TOP keyword). If the NAN keyword is set,
> NANs in the input array are set to 0 in the output array. But 0 falls
> into the valid range for good values (0 to TOP)!
...

How about this non-WHERE approach?
  bb = bytscl(x, ...) + (finite(x) EQ 0)*255b

Craig

--
 ---------- ----------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@REMOVEcow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
 ---------- ----------

Subject: Re: BYTSCL and NAN keyword
Posted by David Fanning on Tue, 02 Mar 2004 17:45:35 GMT
View Forum Message <> Reply to Message

Kenneth Bowman writes:

> Right, I understand all that. I'm just trying to understand what the

> NAN keyword is good for.
>
> If you could say NAN = 255 (or 19, or whatever), instead of just /NAN,
> it would be of some use. That way the value that NANs become in the
> output could be distinguished from valid data.
>
> As it is, the NAN keyword is only of use if you are happy with NANs
> turning into valid data in the output (i.e., indistinguishable from
> small valid values).
>
> I have to consider this an error in the way the handling of NANs was
> implemented in BYTSCL.

Ken, they are *bytes*. They only have 8 little thingamajigs.
There is not too much room for creativity here! :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: BYTSCL and NAN keyword
Posted by JD Smith on Tue, 02 Mar 2004 21:32:00 GMT
View Forum Message <> Reply to Message

On Tue, 02 Mar 2004 10:26:19 -0700, Kenneth Bowman wrote:

> In article <MPG.1aae6fc4ef56f7829896c3@news.frii.com>,
> David Fanning <david@dfanning.com> wrote:
>
>> Well, NANs are definitely floats (that is, they have a float-type
>> bit pattern). So it seems reasonable to me that if you are
>> trying to stuff it into a byte (which can only have values between
>> 0 and 255, that 0 is a good choice. (The only possible other
>> choice is 255, but that just turns your problem on its head.)
>>
>> I think what I would do is locate the NANs and save their
>> indices. Then scale my data into 255 colors (0 to 254), leaving
>> color index 255 for the NAN color (whatever that is). This will
>> take a couple of steps, but that's what IDL programs are for. :-)
>
> Right, I understand all that. I'm just trying to understand what the
> NAN keyword is good for.

>
> If you could say NAN = 255 (or 19, or whatever), instead of just /NAN,
> it would be of some use. That way the value that NANs become in the
> output could be distinguished from valid data.
>
> As it is, the NAN keyword is only of use if you are happy with NANs
> turning into valid data in the output (i.e., indistinguishable from
> small valid values).
>
> I have to consider this an error in the way the handling of NANs was
> implemented in BYTSCL.

```
IDL> print,bytscl([!VALUES.F_NAN,0.,10.,100.])
  0   0   0   0
% Program caused arithmetic error: Floating illegal operand
IDL> print,bytscl([!VALUES.F_NAN,0.,10.,100.],/NAN)
  0   0  25 255
```

What it's good for is ensuring that the rest of your finite values are scaled appropriately. The problem, of course, is that:

```
IDL> print,max([!VALUES.F_NAN,0.,10.,100.])
        NaN
```

so, absent the /NAN, this becomes the maximum value to scale the array to, with the obvious deleterious side effect that all scaled values are now NaN (think of NaN like a virus, infecting everything it touches). Since there's no byte value representing NaN, as there is a float value, they had to pick something, so they picked 0b, which seems as good as anything else. Any value you pick would be degenerate with real data.

If you'd like to separate out the NaN's, I'd use:

```
b=bytscl(a,/NAN,TOP=254)+finite(a)
```

Now 0 is definitely a NaN, and 1-255 is real data. Yes, there is a duplicate check for NaN's implicit here. If you rescale the image many times without changing the data, you might cache finite(a) for a bit of speedup. I even go so far as to check if there are any non-finite values and use:

```
 b=bytscl(a,NAN=self.non_finite,TOP=254)+self.non_finite?self .finite:1
```

The reason? Adding /NaN slows most functions down by about a factor of 2.

JD

Subject: Re: BYTSCL and NAN keyword
Posted by David Fanning on Tue, 02 Mar 2004 22:07:02 GMT
View Forum Message <> Reply to Message

JD Smith writes:

> I even go so far as to check if there are any
> non-finite values and use:
>
> b=bytscl(a,NAN=self.non_finite,TOP=254)+self.non_finite?self .finite:1

There you go! I should have known JD could still
work magic no matter how limited his materials. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: BYTSCL and NAN keyword
Posted by Kenneth P. Bowman on Tue, 02 Mar 2004 23:18:27 GMT
View Forum Message <> Reply to Message

In article <MPG.1aae762bf27e19ee9896c4@news.frii.com>,
David Fanning <david@dfanning.com> wrote:

> Ken, they are *bytes*. They only have 8 little thingamajigs.
> There is not too much room for creativity here! :-)

Sure there are, 2^8 different ways.

Ken

Subject: Re: BYTSCL and NAN keyword
Posted by David Fanning on Tue, 02 Mar 2004 23:48:43 GMT
View Forum Message <> Reply to Message

Kenneth P. Bowman writes:

> Sure there are, 2^8 different ways.

Yes, but all currently used to represent one of the
256 numbers. :-(

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: BYTSCL and NAN keyword
Posted by Kenneth P. Bowman on Wed, 03 Mar 2004 01:44:33 GMT
View Forum Message <> Reply to Message

In article <MPG.1aaecb44b5ab2a789896c6@news.frii.com>,
David Fanning <david@dfanning.com> wrote:

> Yes, but all currently used to represent one of the
> 256 numbers. :-(

You can have the last word.

Ken

Subject: Re: BYTSCL and NAN keyword
Posted by K. Bowman on Wed, 03 Mar 2004 15:26:29 GMT
View Forum Message <> Reply to Message

In article <ond67vf8sb.fsf@cow.physics.wisc.edu>,
Craig Markwardt <craigmnet@REMOVEcow.physics.wisc.edu> wrote:

> How about this non-WHERE approach?
> bb = bytscl(x, ...) + (finite(x) EQ 0)*255b

Clever idea (JD also)! The logical not operator (~) will work as well

```
bb = BYTSCL(x, ..., /NAN) + BYTE(255*(~FINITE(co)))
```

While not mandatory, the BYTE function ensures that the result bb is
type BYTE (since FINITE returns a LONG).

Ken