Subject: Re: Philosophical Question about NAN
Posted by Kenneth P. Bowman on Mon, 17 Nov 2008 15:32:24 GMT
View Forum Message <> Reply to Message

In article <MPG.238b3491ef337cc798a534@news.giganews.com>,
David Fanning <news@dfanning.com> wrote:

> Folks,
>
> I've had a couple of run-ins lately with NANs and I wonder
> why routines like TOTAL and MEAN don't have the NAN keyword
> set to 1 by default. Why does the user have to set it?
>
> I understand the argument that the NAN capability was
> added as an afterthought (or more likely when someone
> standardized the NAN bit pattern), and so the functionality
> was added as an optional addition that enhanced the function
> rather than changed it. But really...is there a reason
> why it is not the default now?
>
> One could argue, I suppose, that having a program stumble
> over a NAN alerts you to its presence in your data. That
> is useful, certainly. But, typically, once I add a NAN
> keyword to my code, I don't know (nor do I or care) if the
> argument has NANs. Is this lazy programming on my part?
>
> I am just wondering whether not setting the default value
> of the NAN keyword to 1 on routines like TOTAL, MEAN,
> et. al is the functional equivalent of not setting the
> default values of the COLOR and BITS_PER_PIXEL keywords
> to the PostScript device to something useful by default.
> That is, an act of negligence on the part of the
> manufacturer.
>
> What say you?
>
> Cheers,
>
> David

HI David,

I think they chose correctly and erred on the side of safety.

If I know there are Nans in my data, I'll take care of it.

If there are Nans in the data that I don't expect, I don't want to
have to set a keyword somewhere to find that out. That is, I don't

want IDL to automatically skip those Nans.

OTOH, I still find this to be frustrating and dangerous

```
IDL> PRINT, TOTAL(REPLICATE(!VALUES.F_NAN, 5), /NAN)
      0.00000
```

There are no valid numbers in the input vector, but TOTAL
returns a valid FLOAT. This makes the NAN keyword useless
in many situations.

Ken

Subject: Re: Philosophical Question about NAN
Posted by wlandsman on Mon, 17 Nov 2008 15:54:44 GMT
View Forum Message <> Reply to Message

On Nov 17, 9:58 am, David Fanning <n...@dfanning.com> wrote:
> Folks,
>
> I've had a couple of run-ins lately with NANs and I wonder
> why routines like TOTAL and MEAN don't have the NAN keyword
> set to 1 by default. Why does the user have to set it?

I agree with the sentiment but also note that always setting /NAN
incurs a non-trivial performance penalty, e.g.

```
 IDL> a = randomn(seed,10000,2000)
 IDL> t = systime(1) & b = total(a) & print,systime(1)-t
      0.25451803
 IDL> t = systime(1) & b = total(a,/nan) & print,systime(1)-t
      0.35278893
```

I've thought at times that arrays should carry a hidden bit saying
whether or not they include NaN values, but this introduces other
overhead problems.

--Wayne

Subject: Re: Philosophical Question about NAN
Posted by Rainer on Mon, 17 Nov 2008 16:01:30 GMT
View Forum Message <> Reply to Message

I wondered about that myself and speculated that checking for NaNs
might decrease performance a little. Never checked this hypothesis,

though.
I often set values deliberately to NaN, so I also need the /NAN
keyword most of the time.

Cheers,
Rainer

Subject: Re: Philosophical Question about NAN
Posted by David Fanning on Mon, 17 Nov 2008 16:02:05 GMT
View Forum Message <> Reply to Message

wlandsman writes:

> I agree with the sentiment but also note that always setting /NAN
> incurs a non-trivial performance penalty, e.g.
>
> IDL> a =3D randomn(seed,10000,2000)
> IDL> t =3D systime(1) & b =3D total(a) & print,systime(1)-t
> 0.25451803
> IDL> t =3D systime(1) & b =3D total(a,/nan) & print,systime(1)-t
> 0.35278893
>
> I've thought at times that arrays should carry a hidden bit saying
> whether or not they include NaN values, but this introduces other
> overhead problems.

I guess I would argue that in the overwhelming number of
cases in my experience, the performance penalty is trivial.
I'm calling these routines a couple of times at most. And
I am not arguing for the elimination of the keyword, only
that the default value could be changed. Thus, if I *was*
experiencing a performance penalty, and I was certain I
had good numbers, I could always set the NAN keyword to 0.

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Philosophical Question about NAN

Posted by wlandsman on Mon, 17 Nov 2008 16:28:27 GMT
View Forum Message <> Reply to Message

On Nov 17, 11:02 am, David Fanning <n...@dfanning.com> wrote:
And
> I am not arguing for the elimination of the keyword, only
> that the default value could be changed. Thus, if I *was*
> experiencing a performance penalty, and I was certain I
> had good numbers, I could always set the NAN keyword to 0.

Yes, I agree with this. I have gotten in the habit of always writing TOTAL(/NAN), but it would be nice if this were the default.

A vaguely related wish of mine is that compile_opt idl2 finally be made the default. I appreciate that ITTVIS wants to have backward compatibility, but square brackets were introduced 11 years ago, and do we still need default 16 bit integers? If there is anyone who still wants their 11 year old software packages to run without modification, ITTVIS could add a compile_opt idl1 (or compile_opt idl_ancient) command.

--Wayne

Subject: Re: Philosophical Question about NAN
Posted by R.Bauer on Tue, 18 Nov 2008 00:38:18 GMT
View Forum Message <> Reply to Message

Sometimes I wish people would use a defined missing value instead on NaN. NaN is only defined for float and double.
If a NaN value is in you data everything can become difficult.

```
IDL> a=[!values.f_nan,0,3,5]
IDL> print,max(a)
       NaN
IDL> print,min(a)
       NaN
IDL> if a[0] gt 1 then print, 'yes' else print, 'no'
no
IDL> if a[0] lt 1 then print, 'yes' else print, 'no'
no
IDL> if a[0] eq 1 then print, 'yes' else print, 'no'
no
```

if you have read until here you may wonder about this
```
IDL> if !values.f_nan eq !values.f_nan then print,'yes' else print, 'no'
no
```

Idl says "no"!!

For functions we can easily set a key so that NaN numbers can be handled
differently but if the default is to search for NaN a lot of other
places needs a lot of changes.

cheers

Reimar

Kenneth P. Bowman schrieb:
> In article <MPG.238b3491ef337cc798a534@news.giganews.com>,
> David Fanning <news@dfanning.com> wrote:
>
>> Folks,
>>
>> I've had a couple of run-ins lately with NANs and I wonder
>> why routines like TOTAL and MEAN don't have the NAN keyword
>> set to 1 by default. Why does the user have to set it?
>>
>> I understand the argument that the NAN capability was
>> added as an afterthought (or more likely when someone
>> standardized the NAN bit pattern), and so the functionality
>> was added as an optional addition that enhanced the function
>> rather than changed it. But really...is there a reason
>> why it is not the default now?
>>
>> One could argue, I suppose, that having a program stumble
>> over a NAN alerts you to its presence in your data. That
>> is useful, certainly. But, typically, once I add a NAN
>> keyword to my code, I don't know (nor do I or care) if the
>> argument has NANs. Is this lazy programming on my part?
>>
>> I am just wondering whether not setting the default value
>> of the NAN keyword to 1 on routines like TOTAL, MEAN,
>> et. al is the functional equivalent of not setting the
>> default values of the COLOR and BITS_PER_PIXEL keywords
>> to the PostScript device to something useful by default.
>> That is, an act of negligence on the part of the
>> manufacturer.
>>
>> What say you?
>>
>> Cheers,
>>
>> David

>
> HI David,
>
> I think they chose correctly and erred on the side of safety.
>
> If I know there are Nans in my data, I'll take care of it.
>
> If there are Nans in the data that I don't expect, I don't want to
> have to set a keyword somewhere to find that out. That is, I don't
> want IDL to automatically skip those Nans.
>
> OTOH, I still find this to be frustrating and dangerous
>
> IDL> PRINT, TOTAL(REPLICATE(!VALUES.F_NAN, 5), /NAN)
> 0.00000
>
> There are no valid numbers in the input vector, but TOTAL
> returns a valid FLOAT. This makes the NAN keyword useless
> in many situations.
>
> Ken

Subject: Re: Philosophical Question about NAN
Posted by pgrigis on Tue, 18 Nov 2008 01:07:12 GMT
View Forum Message <> Reply to Message

On the other hand,
NAN works much better than fixed values for
plots! (for instance, if nan=!values.f_nan
a=[1.0,2,nan,4,2]
will give a much better plot than if nan=-999,
even if one has a good yrange).

Ciao,
Paolo

Reimar Bauer wrote:
> Sometimes I wish people would use a defined missing value instead on
> NaN. NaN is only defined for float and double.
> If a NaN value is in you data everything can become difficult.
>
> IDL> a=[!values.f_nan,0,3,5]
> IDL> print,max(a)
> NaN
> IDL> print,min(a)
> NaN
> IDL> if a[0] gt 1 then print, 'yes' else print, 'no'

> no
> IDL> if a[0] lt 1 then print, 'yes' else print, 'no'
> no
> IDL> if a[0] eq 1 then print, 'yes' else print, 'no'
> no
>
> if you have read until here you may wonder about this
> IDL> if !values.f_nan eq !values.f_nan then print,'yes' else print, 'no'
> no
>
> Idl says "no"!!
>
> For functions we can easily set a key so that NaN numbers can be handled
> differently but if the default is to search for NaN a lot of other
> places needs a lot of changes.
>
> cheers
>
> Reimar
>
>
> Kenneth P. Bowman schrieb:
>> In article <MPG.238b3491ef337cc798a534@news.giganews.com>,
>> David Fanning <news@dfanning.com> wrote:
>>
>>> Folks,
>>>
>>> I've had a couple of run-ins lately with NANs and I wonder
>>> why routines like TOTAL and MEAN don't have the NAN keyword
>>> set to 1 by default. Why does the user have to set it?
>>>
>>> I understand the argument that the NAN capability was
>>> added as an afterthought (or more likely when someone
>>> standardized the NAN bit pattern), and so the functionality
>>> was added as an optional addition that enhanced the function
>>> rather than changed it. But really...is there a reason
>>> why it is not the default now?
>>>
>>> One could argue, I suppose, that having a program stumble
>>> over a NAN alerts you to its presence in your data. That
>>> is useful, certainly. But, typically, once I add a NAN
>>> keyword to my code, I don't know (nor do I or care) if the
>>> argument has NANs. Is this lazy programming on my part?
>>>
>>> I am just wondering whether not setting the default value
>>> of the NAN keyword to 1 on routines like TOTAL, MEAN,
>>> et. al is the functional equivalent of not setting the
>>> default values of the COLOR and BITS_PER_PIXEL keywords

>>> to the PostScript device to something useful by default.
>>> That is, an act of negligence on the part of the
>>> manufacturer.
>>>
>>> What say you?
>>>
>>> Cheers,
>>>
>>> David
>>
>> HI David,
>>
>> I think they chose correctly and erred on the side of safety.
>>
>> If I know there are Nans in my data, I'll take care of it.
>>
>> If there are Nans in the data that I don't expect, I don't want to
>> have to set a keyword somewhere to find that out. That is, I don't
>> want IDL to automatically skip those Nans.
>>
>> OTOH, I still find this to be frustrating and dangerous
>>
>> IDL> PRINT, TOTAL(REPLICATE(!VALUES.F_NAN, 5), /NAN)
>> 0.00000
>>
>> There are no valid numbers in the input vector, but TOTAL
>> returns a valid FLOAT. This makes the NAN keyword useless
>> in many situations.
>>
>> Ken

Subject: Re: Philosophical Question about NAN
Posted by R.Bauer on Tue, 18 Nov 2008 08:18:31 GMT
View Forum Message <> Reply to Message

Paolo schrieb:
> On the other hand,
> NAN works much better than fixed values for
> plots! (for instance, if nan=!values.f_nan
> a=[1.0,2,nan,4,2]
> will give a much better plot than if nan=-999,
> even if one has a good yrange).
>
> Ciao,
> Paolo

the same is true for Inf values

```
inf = 1.0 / 0
a = [1.0, 2, inf, 4,2]
plot, a

print, finite(a)
1   1   0   1   1
```

Just something is possible it does not make it automatically a great
solution.

Reimar

>
> Reimar Bauer wrote:
>> Sometimes I wish people would use a defined missing value instead on
>> NaN. NaN is only defined for float and double.
>> If a NaN value is in you data everything can become difficult.
>>
>> IDL> a=[!values.f_nan,0,3,5]
>> IDL> print,max(a)
>>   NaN
>> IDL> print,min(a)
>>   NaN
>> IDL> if a[0] gt 1 then print, 'yes' else print, 'no'
>> no
>> IDL> if a[0] lt 1 then print, 'yes' else print, 'no'
>> no
>> IDL> if a[0] eq 1 then print, 'yes' else print, 'no'
>> no
>>
>> if you have read until here you may wonder about this
>> IDL> if !values.f_nan eq !values.f_nan then print,'yes' else print, 'no'
>> no
>>
>> Idl says "no"!!
>>
>> For functions we can easily set a key so that NaN numbers can be handled
>> differently but if the default is to search for NaN a lot of other
>> places needs a lot of changes.
>>
>> cheers
>>
>> Reimar
>>

>>
>> Kenneth P. Bowman schrieb:
>>> In article <MPG.238b3491ef337cc798a534@news.giganews.com>,
>>> David Fanning <news@dfanning.com> wrote:
>>>
>>>> Folks,
>>>>
>>>> I've had a couple of run-ins lately with NANs and I wonder
>>>> why routines like TOTAL and MEAN don't have the NAN keyword
>>>> set to 1 by default. Why does the user have to set it?
>>>>
>>>> I understand the argument that the NAN capability was
>>>> added as an afterthought (or more likely when someone
>>>> standardized the NAN bit pattern), and so the functionality
>>>> was added as an optional addition that enhanced the function
>>>> rather than changed it. But really...is there a reason
>>>> why it is not the default now?
>>>>
>>>> One could argue, I suppose, that having a program stumble
>>>> over a NAN alerts you to its presence in your data. That
>>>> is useful, certainly. But, typically, once I add a NAN
>>>> keyword to my code, I don't know (nor do I or care) if the
>>>> argument has NANs. Is this lazy programming on my part?
>>>>
>>>> I am just wondering whether not setting the default value
>>>> of the NAN keyword to 1 on routines like TOTAL, MEAN,
>>>> et. al is the functional equivalent of not setting the
>>>> default values of the COLOR and BITS_PER_PIXEL keywords
>>>> to the PostScript device to something useful by default.
>>>> That is, an act of negligence on the part of the
>>>> manufacturer.
>>>>
>>>> What say you?
>>>>
>>>> Cheers,
>>>>
>>>> David
>>> HI David,
>>>
>>> I think they chose correctly and erred on the side of safety.
>>>
>>> If I know there are Nans in my data, I'll take care of it.
>>>
>>> If there are Nans in the data that I don't expect, I don't want to
>>> have to set a keyword somewhere to find that out. That is, I don't
>>> want IDL to automatically skip those Nans.
>>>
>>> OTOH, I still find this to be frustrating and dangerous

>>>
>>> IDL> PRINT, TOTAL(REPLICATE(!VALUES.F_NAN, 5), /NAN)
>>>   0.00000
>>>
>>> There are no valid numbers in the input vector, but TOTAL
>>> returns a valid FLOAT. This makes the NAN keyword useless
>>> in many situations.
>>>
>>> Ken

Subject: Re: Philosophical Question about NAN
Posted by Jeremy Bailin on Tue, 18 Nov 2008 13:05:58 GMT
View Forum Message <> Reply to Message

On Nov 17, 9:58 am, David Fanning <n...@dfanning.com> wrote:
> Folks,
>
> I've had a couple of run-ins lately with NANs and I wonder
> why routines like TOTAL and MEAN don't have the NAN keyword
> set to 1 by default. Why does the user have to set it?
>
> I understand the argument that the NAN capability was
> added as an afterthought (or more likely when someone
> standardized the NAN bit pattern), and so the functionality
> was added as an optional addition that enhanced the function
> rather than changed it. But really...is there a reason
> why it is not the default now?
>
> One could argue, I suppose, that having a program stumble
> over a NAN alerts you to its presence in your data. That
> is useful, certainly. But, typically, once I add a NAN
> keyword to my code, I don't know (nor do I or care) if the
> argument has NANs. Is this lazy programming on my part?
>
> I am just wondering whether not setting the default value
> of the NAN keyword to 1 on routines like TOTAL, MEAN,
> et. al is the functional equivalent of not setting the
> default values of the COLOR and BITS_PER_PIXEL keywords
> to the PostScript device to something useful by default.
> That is, an act of negligence on the part of the
> manufacturer.
>
> What say you?
>
> Cheers,
>
> David

> --
> David Fanning, Ph.D.
> Fanning Software Consulting, Inc.
> Coyote's Guide to IDL Programming:http://www.dfanning.com/
> Sepore ma de ni thui. ("Perhaps thou speakest truth.")

My 2 cents... is that about 75% of the time that my data ends up
having NaNs in it, it's not intentional and is a sign of something
screwy. So by not enabling /NAN by default, debugging becomes much
simpler - it's immediately obvious if the result is NaN that
something's gone wrong, while it's not obvious if it gives me some
real but incorrect number.

-Jeremy.

Subject: Re: Philosophical Question about NAN
Posted by pgrigis on Tue, 18 Nov 2008 16:42:44 GMT
View Forum Message <> Reply to Message

Reimar Bauer wrote:
> Paolo schrieb:
>> On the other hand,
>> NAN works much better than fixed values for
>> plots! (for instance, if nan=!values.f_nan
>> a=[1.0,2,nan,4,2]
>> will give a much better plot than if nan=-999,
>> even if one has a good yrange).
>>
>> Ciao,
>> Paolo
>
>
> the same is true for Inf values

Well, when I need to plot data with missing
values, I put in NANs in my array. If I wouldn't,
I would have to loop over the valid data chuncks
to do a nice plot...now, we don't want to do that,
do we? So I hold on to my point...

Ciao,
Paolo

>
> inf = 1.0 / 0
> a = [1.0, 2, inf, 4,2]

> plot, a
>
> print, finite(a)
> 1 1 0 1 1
>
> Just something is possible it does not make it automatically a great
> solution.
>
>
> Reimar
>
>
>>
>> Reimar Bauer wrote:
>>> Sometimes I wish people would use a defined missing value instead on
>>> NaN. NaN is only defined for float and double.
>>> If a NaN value is in you data everything can become difficult.
>>>
>>> IDL> a=[!values.f_nan,0,3,5]
>>> IDL> print,max(a)
>>> NaN
>>> IDL> print,min(a)
>>> NaN
>>> IDL> if a[0] gt 1 then print, 'yes' else print, 'no'
>>> no
>>> IDL> if a[0] lt 1 then print, 'yes' else print, 'no'
>>> no
>>> IDL> if a[0] eq 1 then print, 'yes' else print, 'no'
>>> no
>>>
>>> if you have read until here you may wonder about this
>>> IDL> if !values.f_nan eq !values.f_nan then print,'yes' else print, 'no'
>>> no
>>>
>>> Idl says "no"!!
>>>
>>> For functions we can easily set a key so that NaN numbers can be handled
>>> differently but if the default is to search for NaN a lot of other
>>> places needs a lot of changes.
>>>
>>> cheers
>>>
>>> Reimar
>>>
>>>
>>> Kenneth P. Bowman schrieb:
>>>> In article <MPG.238b3491ef337cc798a534@news.giganews.com>,
>>>> David Fanning <news@dfanning.com> wrote:

>>>>
>>>> > Folks,
>>>> >
>>>> > I've had a couple of run-ins lately with NANs and I wonder
>>>> > why routines like TOTAL and MEAN don't have the NAN keyword
>>>> > set to 1 by default. Why does the user have to set it?
>>>> >
>>>> > I understand the argument that the NAN capability was
>>>> > added as an afterthought (or more likely when someone
>>>> > standardized the NAN bit pattern), and so the functionality
>>>> > was added as an optional addition that enhanced the function
>>>> > rather than changed it. But really...is there a reason
>>>> > why it is not the default now?
>>>> >
>>>> > One could argue, I suppose, that having a program stumble
>>>> > over a NAN alerts you to its presence in your data. That
>>>> > is useful, certainly. But, typically, once I add a NAN
>>>> > keyword to my code, I don't know (nor do I or care) if the
>>>> > argument has NANs. Is this lazy programming on my part?
>>>> >
>>>> > I am just wondering whether not setting the default value
>>>> > of the NAN keyword to 1 on routines like TOTAL, MEAN,
>>>> > et. al is the functional equivalent of not setting the
>>>> > default values of the COLOR and BITS_PER_PIXEL keywords
>>>> > to the PostScript device to something useful by default.
>>>> > That is, an act of negligence on the part of the
>>>> > manufacturer.
>>>> >
>>>> > What say you?
>>>> >
>>>> > Cheers,
>>>> >
>>>> > David
>>>> HI David,
>>>>
>>>> I think they chose correctly and erred on the side of safety.
>>>>
>>>> If I know there are Nans in my data, I'll take care of it.
>>>>
>>>> If there are Nans in the data that I don't expect, I don't want to
>>>> have to set a keyword somewhere to find that out. That is, I don't
>>>> want IDL to automatically skip those Nans.
>>>>
>>>> OTOH, I still find this to be frustrating and dangerous
>>>>
>>>> IDL> PRINT, TOTAL(REPLICATE(!VALUES.F_NAN, 5), /NAN)
>>>>   0.00000
>>>>

>>>> There are no valid numbers in the input vector, but TOTAL
>>>> returns a valid FLOAT. This makes the NAN keyword useless
>>>> in many situations.
>>>>
>>>> Ken

Subject: Re: Philosophical Question about NAN
Posted by R.Bauer on Tue, 18 Nov 2008 20:55:31 GMT
View Forum Message <> Reply to Message

Paolo schrieb:
>
> Reimar Bauer wrote:
>> Paolo schrieb:
>>> On the other hand,
>>> NAN works much better than fixed values for
>>> plots! (for instance, if nan=!values.f_nan
>>> a=[1.0,2,nan,4,2]
>>> will give a much better plot than if nan=-999,
>>> even if one has a good yrange).
>>>
>>> Ciao,
>>> Paolo
>>
>> the same is true for Inf values
>
> Well, when I need to plot data with missing
> values, I put in NANs in my array. If I wouldn't,
> I would have to loop over the valid data chuncks
> to do a nice plot...now, we don't want to do that,
> do we? So I hold on to my point...
>
> Ciao,
> Paolo
>

I do understand your point,
we have tons of finite based routines in our library.

e.g.
http://www.fz-juelich.de/icg/icg-1/idl_icglib/idl_source/idl _work/fh_lib/f_eq.pro

My point is that some routines do need a keyword and others not.
However the default is it should not be mixed up.

And plot should not behave exactly the same for Inf and NaN data. But it
does. So the result is not well defined.

Reimar

>
>> inf = 1.0 / 0
>> a = [1.0, 2, inf, 4,2]
>> plot, a
>>
>> print, finite(a)
>> 1 1 0 1 1
>>
>> Just something is possible it does not make it automatically a great
>> solution.
>>
>>
>> Reimar
>>
>>
>>> Reimar Bauer wrote:
>>>> Sometimes I wish people would use a defined missing value instead on
>>>> NaN. NaN is only defined for float and double.
>>>> If a NaN value is in you data everything can become difficult.
>>>>
>>>> IDL> a=[!values.f_nan,0,3,5]
>>>> IDL> print,max(a)
>>>> NaN
>>>> IDL> print,min(a)
>>>> NaN
>>>> IDL> if a[0] gt 1 then print, 'yes' else print, 'no'
>>>> no
>>>> IDL> if a[0] lt 1 then print, 'yes' else print, 'no'
>>>> no
>>>> IDL> if a[0] eq 1 then print, 'yes' else print, 'no'
>>>> no
>>>>
>>>> if you have read until here you may wonder about this
>>>> IDL> if !values.f_nan eq !values.f_nan then print,'yes' else print, 'no'
>>>> no
>>>>
>>>> Idl says "no"!!
>>>>
>>>> For functions we can easily set a key so that NaN numbers can be handled
>>>> differently but if the default is to search for NaN a lot of other
>>>> places needs a lot of changes.
>>>>
>>>> cheers
>>>>

>>>> Reimar
>>>>
>>>>
>>>> Kenneth P. Bowman schrieb:
>>>> > In article <MPG.238b3491ef337cc798a534@news.giganews.com>,
>>>> > David Fanning <news@dfanning.com> wrote:
>>>> >
>>>> >> Folks,
>>>> >>
>>>> >> I've had a couple of run-ins lately with NANs and I wonder
>>>> >> why routines like TOTAL and MEAN don't have the NAN keyword
>>>> >> set to 1 by default. Why does the user have to set it?
>>>> >>
>>>> >> I understand the argument that the NAN capability was
>>>> >> added as an afterthought (or more likely when someone
>>>> >> standardized the NAN bit pattern), and so the functionality
>>>> >> was added as an optional addition that enhanced the function
>>>> >> rather than changed it. But really...is there a reason
>>>> >> why it is not the default now?
>>>> >>
>>>> >> One could argue, I suppose, that having a program stumble
>>>> >> over a NAN alerts you to its presence in your data. That
>>>> >> is useful, certainly. But, typically, once I add a NAN
>>>> >> keyword to my code, I don't know (nor do I or care) if the
>>>> >> argument has NANs. Is this lazy programming on my part?
>>>> >>
>>>> >> I am just wondering whether not setting the default value
>>>> >> of the NAN keyword to 1 on routines like TOTAL, MEAN,
>>>> >> et. al is the functional equivalent of not setting the
>>>> >> default values of the COLOR and BITS_PER_PIXEL keywords
>>>> >> to the PostScript device to something useful by default.
>>>> >> That is, an act of negligence on the part of the
>>>> >> manufacturer.
>>>> >>
>>>> >> What say you?
>>>> >>
>>>> >> Cheers,
>>>> >>
>>>> >> David
>>>> > HI David,
>>>> >
>>>> > I think they chose correctly and erred on the side of safety.
>>>> >
>>>> > If I know there are Nans in my data, I'll take care of it.
>>>> >
>>>> > If there are Nans in the data that I don't expect, I don't want to
>>>> > have to set a keyword somewhere to find that out. That is, I don't
>>>> > want IDL to automatically skip those Nans.

>>>> >
>>>> > OTOH, I still find this to be frustrating and dangerous
>>>> >
>>>> > IDL> PRINT, TOTAL(REPLICATE(!VALUES.F_NAN, 5), /NAN)
>>>> >   0.00000
>>>> >
>>>> > There are no valid numbers in the input vector, but TOTAL
>>>> > returns a valid FLOAT. This makes the NAN keyword useless
>>>> > in many situations.
>>>> >
>>>> > Ken

Subject: Re: Philosophical Question about NAN
Posted by R.G. Stockwell on Wed, 19 Nov 2008 14:32:43 GMT
View Forum Message <> Reply to Message

"David Fanning" <news@dfanning.com> wrote in message
news:MPG.238b3491ef337cc798a534@news.giganews.com...
> Folks,
>
> I've had a couple of run-ins lately with NANs and I wonder
> why routines like TOTAL and MEAN don't have the NAN keyword
> set to 1 by default. Why does the user have to set it?

My two cents. In spectral analysis, a 'missing point' or NAN has
a profound effect. The FFT _must_ have the behaviour it currently
has (it returns all nans, if there are any nans).

The position/time of each
measurement is critical in an FFT, and you can't just throw a number away
and translate all the other points.

Also, it is not clear on how to handle a NAN - it is ambiguous.
Does one throw the point away (like in a MEAN() calculation), or
does one interpolate the data (like in an FFT() application)?
Automated interpolation can be a very bad idea (and what happens
if the NAN results in an extrapolation, cause there are no surrounding
points?) Imho, the programmer has to make the decision about how to
handle NANs.

Also, where data is missing can be important even in functions like MEAN().
Suppose you have a year of temperature data taken every hour, but in winter,
you only have daytime data (i.e. NANs at night).
If you take monthly data, you will get the a very wrong result in terms of
the

annual variation of monthly means, and it will be artificial, and you won't know about it.

cheers,
bob

Subject: Re: Philosophical Question about NAN
Posted by Foldy Lajos on Wed, 19 Nov 2008 16:09:09 GMT
View Forum Message <> Reply to Message

On Wed, 19 Nov 2008, R.G. Stockwell wrote:

> Also, it is not clear on how to handle a NAN - it is ambiguous.

Even IDL is confused. For a single element input, /cumul in TOTAL should be a no-op. Let's try it:

```
IDL> print, !version
{ x86_64 linux unix linux 7.0 Oct 25 2007      64      64}
IDL> a=complex(1.0, !values.f_nan)
IDL> print, total(a, /nan)
(      0.00000,      0.00000)
IDL> print, total(a, /nan, /cumul)
(      1.00000,      0.00000)
```

regards,
lajos

Subject: Re: Philosophical Question about NAN
Posted by Mark[1] on Thu, 20 Nov 2008 00:24:54 GMT
View Forum Message <> Reply to Message

I think backward compatibility should be the overriding principle here. So the NAN keyword defaults to unset and COMPILE_OPT IDL2 remains necessary for the new syntax. There's simply too much old code around that shouldn't be broken unless absolutely necessary.

(Mind you, I pay very little attention to that with my own code, but it's my code so I can do what I want!)
