Subject: Re: rebin and !values.f_nan
Posted by David Fanning on Sun, 15 Jul 2007 16:20:34 GMT
View Forum Message <> Reply to Message

Nick writes:

> I have regular arrayed data (1440*720). I'd like to change this data
> to 360*180 array.
> So I use 'Rebin' function.
> But these data have NaN value.
> If I use Rebin and there is a NaN value, new array becomes also NaN.
>
> For example, if there is only one NaN in the old array, the new-array
> becomes NaN. But I want to make a new array except NaN data. This
> situation makes residual data wasteful.
>
> A = [1.5,2.5,3.6,4,7,8.8,9.0,!values.f_nan]
> print, rebin(A, 1)
> ;result is 'NAN'
> ;That I expected value is mean(A, /nan)
>
> Is there any know-how to change array except NaN?

Well, you seem to know how to change your array.
Why don't you just find the NANs, change them to
what you want them to be, then do the REBIN?

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: rebin and !values.f_nan
Posted by Nick[1] on Sun, 15 Jul 2007 21:20:01 GMT
View Forum Message <> Reply to Message

Thank you for your reply.
I could find where is 'NaN' vaule, but I had to omit these 'NaN' value
in order to change array. It's hard point for me. If I change NaN
values to zero, values which are changed by Rebin will be lower than
original values (True values).

A = [1.5,2.5,3.6,4,7,8.8,9.0,!values.f_nan]

print, rebin(A, 1)
;True value is 5.2
;But if I changed NaN to zero, the result is 4.55

The example case is so simple that I can fix it easily, but my data
have 1440*720 array.
So I couldn't fix thise one by one. Is there any methods?
Thanks, Nick.

Subject: Re: rebin and !values.f_nan
Posted by David Fanning on Sun, 15 Jul 2007 22:13:48 GMT
View Forum Message <> Reply to Message

Nick writes:

> Thank you for your reply.
> I could find where is 'NaN' vaule, but I had to omit these 'NaN' value
> in order to change array. It's hard point for me. If I change NaN
> values to zero, values which are changed by Rebin will be lower than
> original values (True values).
>
> A = [1.5,2.5,3.6,4,7,8.8,9.0,!values.f_nan]
> print, rebin(A, 1)
> ;True value is 5.2
> ;But if I changed NaN to zero, the result is 4.55
>
> The example case is so simple that I can fix it easily, but my data
> have 1440*720 array.
> So I couldn't fix thise one by one. Is there any methods?

I suppose there could be an infinite number of "methods,"
but the point is that all of them would create "data" out
of whole cloth. Thus, we are going to leave the responsibility
for it up to you, who has to answer for it in front of a
scientific audience. (You have told your advisor what you
are up to, I presume.)

How many NANs are there? So many you can't fix them!?
Perhaps your time could be spent more profitably trying
to figure out how to collect more real data. :-)

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: rebin and !values.f_nan
Posted by James Kuyper on Mon, 16 Jul 2007 01:42:32 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> Nick writes:
>
>> I have regular arrayed data (1440*720). I'd like to change this data
>> to 360*180 array.
>> So I use 'Rebin' function.
>> But these data have NaN value.
>> If I use Rebin and there is a NaN value, new array becomes also NaN.
>>
>> For example, if there is only one NaN in the old array, the new-array
>> becomes NaN. But I want to make a new array except NaN data. This
>> situation makes residual data wasteful.
>>
>> A = [1.5,2.5,3.6,4,7,8.8,9.0,!values.f_nan]
>> print, rebin(A, 1)
>> ;result is 'NAN'
>> ;That I expected value is mean(A, /nan)
>>
>> Is there any know-how to change array except NaN?
>
> Well, you seem to know how to change your array.
> Why don't you just find the NANs, change them to
> what you want them to be, then do the REBIN?

I haven't had to do this with IDL, so I didn't realize that IDL handled it inconveniently. What I would normally want to have happen when re-binning data with NaNs is that every element of the output array whose calculation involved one of the NaN's in the input array would itself contain a NaN, while all the other elements of the array would be calculated normally. That's not something that could be achieved by the approach you suggest. If that's a feature not already provided as an option by rebin, it should be.

Subject: Re: rebin and !values.f_nan
Posted by David Fanning on Mon, 16 Jul 2007 01:59:00 GMT
View Forum Message <> Reply to Message

kuyper writes:

> I haven't had to do this with IDL, so I didn't realize that IDL
> handled it inconveniently. What I would normally want to have happen
> when re-binning data with NaNs is that every element of the output
> array whose calculation involved one of the NaN's in the input array
> would itself contain a NaN, while all the other elements of the array
> would be calculated normally. That's not something that could be
> achieved by the approach you suggest. If that's a feature not already
> provided as an option by rebin, it should be.

In my little experiment, that is *exactly* how it
worked. And how I would have expected it to work, too.
How could IDL do anything else and not be accused of
"created data where none exists."

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: rebin and !values.f_nan
Posted by James Kuyper on Mon, 16 Jul 2007 04:04:35 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> kuyper writes:
>
>> I haven't had to do this with IDL, so I didn't realize that IDL
>> handled it inconveniently. What I would normally want to have happen
>> when re-binning data with NaNs is that every element of the output
>> array whose calculation involved one of the NaN's in the input array
>> would itself contain a NaN, while all the other elements of the array
>> would be calculated normally. That's not something that could be
>> achieved by the approach you suggest. If that's a feature not already
>> provided as an option by rebin, it should be.
>
> In my little experiment, that is *exactly* how it
> worked. And how I would have expected it to work, too.
> How could IDL do anything else and not be accused of
> "created data where none exists."

I've been at home this weekend, without access to IDL, so I hadn't

gotten around to testing it. However, what Nick diescribed was not the
creation of data where none exists, but the destruction of usable data
by rebin():

> If I use Rebin and there is a NaN value, new array becomes also NaN.

I understood that to mean that the entire re-binned array was set to
NaN, not just isolated portions of it.

Subject: Re: rebin and !values.f_nan
Posted by David Fanning on Mon, 16 Jul 2007 04:17:57 GMT
View Forum Message <> Reply to Message

kuyper writes:

> I've been at home this weekend, without access to IDL, so I hadn't
> gotten around to testing it. However, what Nick diescribed was not the
> creation of data where none exists, but the destruction of usable data
> by rebin():
>
>> If I use Rebin and there is a NaN value, new array becomes also NaN.
>
> I understood that to mean that the entire re-binned array was set to
> NaN, not just isolated portions of it.

His example showed an entire vector, one element of which was
a NAN, reduced to a single value. In this case, the result--
sensibly I think, since it WAS involved in the calculation--
was a NAN. In the example, the mean was probably a better
choice for a single number (and you can set a NAN flag for
that), but for some reason the requester rejected that as
an option. (He didn't explain why.)

My only point is that if you are going to assign a
value where one didn't exist before, you will have to
take responsibility for doing so. IDL can hardly be
expected to help you out in this ethically challenging
situation. I think it is right that any expression that
involves a NAN will result in a NAN. What else could
it be. :-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.

Subject: Re: rebin and !values.f_nan
Posted by Dick Jackson on Mon, 16 Jul 2007 07:03:19 GMT
View Forum Message <> Reply to Message

Hi all,

"Nick" <jungbinusu@hotmail.com> wrote in message
news:1184534401.344733.293590@i38g2000prf.googlegroups.com.. .
> Thank you for your reply.
> I could find where is 'NaN' vaule, but I had to omit these 'NaN' value
> in order to change array. It's hard point for me. If I change NaN
> values to zero, values which are changed by Rebin will be lower than
> original values (True values).
>
> A = [1.5,2.5,3.6,4,7,8.8,9.0,!values.f_nan]
> print, rebin(A, 1)
> ;True value is 5.2
> ;But if I changed NaN to zero, the result is 4.55
>
> The example case is so simple that I can fix it easily, but my data
> have 1440*720 array.
> So I couldn't fix thise one by one. Is there any methods?
> Thanks, Nick.

Just to clear up one possible confusion, IDL's Rebin() does handle each resulting bin separately, so any bin with no NaNs comes up fine:

```
IDL> A = [1.5,2.5,3.6,4,7,8.8,9.0,!values.f_nan]
IDL> print,rebin(a,4)
2.00000 3.80000 7.90000 NaN
```

Rebin() gives a NaN result if any value in the bin is NaN. Now, what I think Nick wants is for each bin to receive the mean of the finite values (the non-NaNs, or NaNaNs! or perhaps Ns... :-) that are present. A reasonable request, in the spirit of Mean(array, /NaN). I don't think there's a built-in way to do this NaN-tolerant Rebin, so here's my attempt.

I take the first dimension (1440) and split it into two dimensions (4, 360). I take the second dimension (720) and split it into two more dimensions (4, 180). Then I use Total(/NaN) to squash out the extra dimensions, getting the total of the desired values in each resulting array element. A similar Total-ing on the count of Finite() values gives the count of values summed for each result. Divide each total by each count and you get the mean of each bin's finite values. If a bin had all NaNs, the result should be NaN as well.

```
PRO RebinNaNTest

a = FIndGen(1440,720)                  ; Sample data

a[1] = !values.f_nan                   ; to make one element NaN
;a[*,0:2] = !values.f_nan               ; to make three rows NaN
;a[*,0:3] = !values.f_nan               ; to make four rows NaN

Print, 'Rebin method:'

rebinResult = Rebin(a,360,180)

Print, rebinResult[0:1, 0:1]

b = Reform(a, 4, 360, 4, 180)          ; Make separate 'b' array in
                                       ; case you want to see this:
;print,total(a[0:3,0:3])
;          NaN
;print,total(a[0:3,0:3],/NaN)
;      34583.0
;print,total(b[*,0,*,0],/NaN)
;      34583.0
;;    But in practice, you could reform 'a' in place,
;;    which saves memory and is very fast:
;;    a = Reform(a, 4, 360, 4, 180, /Overwrite)

Print
Print, 'RebinNaN method:'

sumFinite = Total(Total(b, 3, /NaN), 1, /NaN)
nFinite = Total(Total(Finite(b), 3, /NaN), 1, /NaN)
result = sumFinite/nFinite

Print, result[0:1, 0:1]

END
```

With one NaN this seems to work:

```
Rebin method:
         NaN      2165.50
     7921.50      7925.50
```

RebinNaN method:
    2305.53   2165.50
    7921.50   7925.50

With three rows of NaNs:

IDL> rebinnantest
Rebin method:
       NaN   NaN
    7921.50   7925.50

RebinNaN method:
    4321.50   4325.50
    7921.50   7925.50

With four rows of NaNs:

IDL> rebinnantest
Rebin method:
       NaN   NaN
    7921.50   7925.50

RebinNaN method:
      -NaN   -NaN
    7921.50   7925.50
Program caused arithmetic error: Floating illegal operand

Rough timing test shows this takes 5-10 times as long as Rebin(), but it works!

This could be put into a nice general function to replace Rebin when you need
NaN handling, but I'll see whether anyone pops up Monday morning with one
already written!

Hope this helps.

--
Cheers,
-Dick

--

Dick Jackson Software Consulting http://www.d-jackson.com
Victoria, BC, Canada +1-250-220-6117 dick@d-jackson.com