Subject: avoiding "floating illegal operand" errors with /nan keyword in mean
Posted by Paul Levine on Tue, 20 Aug 2013 23:37:29 GMT
View Forum Message <> Reply to Message

I have an array with 2800 rows, 2800 columns, and 120 layers, where each layer is a month from a 10-year time series). I would like to calculate annual means, so I will end up with an array that is 2800 x 2800 x 10. So

```
for j = 0, 9 do begin
 ; make one year subset of 2800x1800x12
 subset = array[*,*,j*12:j*12+12]
 newarray[0,0,j] = mean(newarray, dimension=3, /nan)
endfor
```

I am using the /nan keyword because there are a lot of NaNs in the data. As a result, I get
% Program caused arithmetic error: Floating illegal operand
whenever the mean function tries to calculate an average completely out of NaN values.

I know that I could just ignore the error, because the results are what I want them to be, but I'm sure it would be better to figure out how to prevent the error.

So, I tried the following method of checking to see whether I'm dealing with all NaNs

```
for j = 0, 9 do begin
 ; make one year subset of 2800x1800x12
 subset = array[*,*,j*12:j*12+12]
 if max(finite(subset)) eq 1 then begin
  newarray[0,0,j] = mean(newarray, dimension=3, /nan)
 endif else begin
  newarray[0,0,j] = make_array(2800,2800,value=!VALUES.D_NAN)
 endelse
endfor
```

This eliminates the error for any situation where the entire subset is NaN. However, because the mean function is essentially calculating 2800x2800 means of 12 elements each, there are instances where only one or a few of those 12-element sets are completely NaN, which is not caught by my method of checking, so I still get the error message.

The only way I can imagine extending my checking method would be to loop through every row and column of the data set, calculating each mean one at a time so that I can check whether or not all 12 elements are NaN. Of course, that would be incredibly inefficient, so I would

not seriously entertain that idea.

Is there a better solution out there, or should I just suck it up and
live with the error message?

Subject: Re: avoiding "floating illegal operand" errors with /nan keyword in mean
Posted by wlandsman on Wed, 21 Aug 2013 00:26:02 GMT
View Forum Message <> Reply to Message

Does setting !except=0 work for you? --Wayne

On Tuesday, August 20, 2013 7:37:29 PM UTC-4, Paul Levine wrote:
> I have an array with 2800 rows, 2800 columns, and 120 layers, where
>
> each layer is a month from a 10-year time series). I would like to
>
> calculate annual means, so I will end up with an array that is 2800 x
>
> 2800 x 10. So
>
>
>
> for j = 0, 9 do begin
>
> ; make one year subset of 2800x1800x12
>
> subset = array[*,*,j*12:j*12+12]
>
> newarray[0,0,j] = mean(newarray, dimension=3, /nan)
>
> endfor
>
>
>
> I am using the /nan keyword because there are a lot of NaNs in the
>
> data. As a result, I get
>
> % Program caused arithmetic error: Floating illegal operand
>
> whenever the mean function tries to calculate an average completely out
>
> of NaN values.
>
>
>
> I know that I could just ignore the error, because the results are what

>
> I want them to be, but I'm sure it would be better to figure out how to
>
> prevent the error.
>
>
>
> So, I tried the following method of checking to see whether I'm dealing
>
> with all NaNs
>
>
>
> for j = 0, 9 do begin
>
> ; make one year subset of 2800x1800x12
>
> subset = array[*,*,j*12:j*12+12]
>
> if max(finite(subset)) eq 1 then begin
>
>   newarray[0,0,j] = mean(newarray, dimension=3, /nan)
>
> endif else begin
>
>   newarray[0,0,j] = make_array(2800,2800,value=!VALUES.D_NAN)
>
> endelse
>
> endfor
>
>
>
> This eliminates the error for any situation where the entire subset is
>
> NaN. However, because the mean function is essentially calculating
>
> 2800x2800 means of 12 elements each, there are instances where only one
>
> or a few of those 12-element sets are completely NaN, which is not
>
> caught by my method of checking, so I still get the error message.
>
>
>
> The only way I can imagine extending my checking method would be to
>
> loop through every row and column of the data set, calculating each

>
> mean one at a time so that I can check whether or not all 12 elements
>
> are NaN. Of course, that would be incredibly inefficient, so I would
>
> not seriously entertain that idea.
>
>
>
> Is there a better solution out there, or should I just suck it up and
>
> live with the error message?

Subject: Re: avoiding "floating illegal operand" errors with /nan keyword in mean
Posted by Paul Levine on Wed, 21 Aug 2013 02:17:41 GMT
View Forum Message <> Reply to Message

On 2013-08-21 00:26:02 +0000, wlandsman said:

> Does setting !except=0 work for you? --Wayne

Thank you for the suggestion. It does work insofar as the error message does not appear, though when I check_math I get 128, so isn't that more or less the equivalent of simply ignoring the errors?

I realized that the example code I banged out in my initial post had a couple typos; here is some double-checked code that presents the problem I am having, guaranteed to run properly.

The example data is completely contrived, though it essentially a much smaller version of the data I am dealing with (it simulates 9 "images" with 3 columns and 3 rows where I want to average every 3 images).

In example 1, my method of checking works, and the math error is avoided, because in the second iteration of the loop, the entire subset is NaN so max(finite(subset)) is zero. The only difference between examples 1 and 2 is in the second line, which sets certain elements of the array to NaN. In example 2, in the second iteration of the loop only one of the "pixels" is all NaN, which my method of checking does not catch, thus yielding the math error.

```
; Example 1
array = findgen(3,3,9)
array[*,*,3:5] = !VALUES.D_NAN
```

```
newarray = fltarr(3,3,3)
for j = 0, 2 do begin
  subset = array[*,*,j*3:j*3+2]
  if max(finite(subset)) eq 1 then begin
    newarray[0,0,j] = mean(subset, dimension=3, /nan)
  endif else begin
    newarray[0,0,j] = make_array(3,3,value=!VALUES.D_NAN)
    print, 'else case happens'
  endelse
endfor

; Example 2
array = findgen(3,3,9)
array[2,2,3:5] = !VALUES.D_NAN
newarray = fltarr(3,3,3)
for j = 0, 2 do begin
  subset = array[*,*,j*3:j*3+2]
  if max(finite(subset)) eq 1 then begin
    newarray[0,0,j] = mean(subset, dimension=3, /nan)
  endif else begin
    newarray[0,0,j] = make_array(3,3,value=!VALUES.D_NAN)
    print, 'else case happens'
  endelse
endfor
```

Subject: Re: avoiding "floating illegal operand" errors with /nan keyword in mean
Posted by wlandsman on Wed, 21 Aug 2013 12:58:38 GMT
View Forum Message <> Reply to Message

On Tuesday, August 20, 2013 10:17:41 PM UTC-4, Paul Levine wrote:
> On 2013-08-21 00:26:02 +0000, wlandsman said:
>
>
>> Does setting !except=0 work for you? --Wayne
>
> Thank you for the suggestion. It does work insofar as the error
> message does not appear, though when I check_math I get 128, so isn't
> that more or less the equivalent of simply ignoring the errors?
>

Yes, setting !EXCEPT suppresses the annoying messages

Why don't you want to ignore the errors (really just warnings)? It doesn't hurt the computer to compute floating illegal operands. You did say you were interested in efficiency and I'm fairly certain that ignoring the warning messages would be the fastest method. Just be sure to reset !EXCEPT afterwards so that you catch errors/warnings in the rest of your code. If you want to be extra careful then make sure the CHECK_MATH output is 128 (floating illegal operand) and

no other math error occurred durning the mean() calculation.

I know this doesn't answer the question you posted, which is an interesting problem in its own right. But sometimes the best way to untie a knot is to cut it with a knife. --Wayne

---

Subject: Re: avoiding "floating illegal operand" errors with /nan keyword in mean
Posted by Fabzi on Wed, 21 Aug 2013 21:11:45 GMT
View Forum Message <> Reply to Message

Hi Paul,

I also had the same problem:
https://groups.google.com/forum/#!topic/comp.lang.idl-pvwave /XmPXtQE6VZ0

It is not necessary to use a for loop to avoid the warnings, you just need to write your own MEAN() where you check for cases with TOTAL(FINITE(data), dimension) eq 0 but you have to decide if it worth it or not...

On 08/21/2013 01:37 AM, Paul Levine wrote:
> I have an array with 2800 rows, 2800 columns, and 120 layers, where each
> layer is a month from a 10-year time series). I would like to calculate
> annual means, so I will end up with an array that is 2800 x 2800 x 10. So
>
> for j = 0, 9 do begin
> ; make one year subset of 2800x1800x12
> subset = array[*,*,j*12:j*12+12]
> newarray[0,0,j] = mean(newarray, dimension=3, /nan)
> endfor
>
> I am using the /nan keyword because there are a lot of NaNs in the
> data. As a result, I get
> % Program caused arithmetic error: Floating illegal operand
> whenever the mean function tries to calculate an average completely out
> of NaN values.
>
> I know that I could just ignore the error, because the results are what
> I want them to be, but I'm sure it would be better to figure out how to
> prevent the error.
>
> So, I tried the following method of checking to see whether I'm dealing
> with all NaNs
>
> for j = 0, 9 do begin
> ; make one year subset of 2800x1800x12
> subset = array[*,*,j*12:j*12+12]

> if max(finite(subset)) eq 1 then begin
> newarray[0,0,j] = mean(newarray, dimension=3, /nan)
> endif else begin
> newarray[0,0,j] = make_array(2800,2800,value=!VALUES.D_NAN)
> endelse
> endfor
>
> This eliminates the error for any situation where the entire subset is
> NaN. However, because the mean function is essentially calculating
> 2800x2800 means of 12 elements each, there are instances where only one
> or a few of those 12-element sets are completely NaN, which is not
> caught by my method of checking, so I still get the error message.
>
> The only way I can imagine extending my checking method would be to loop
> through every row and column of the data set, calculating each mean one
> at a time so that I can check whether or not all 12 elements are NaN.
> Of course, that would be incredibly inefficient, so I would not
> seriously entertain that idea.
>
> Is there a better solution out there, or should I just suck it up and
> live with the error message?
>

Subject: Re: avoiding "floating illegal operand" errors with /nan keyword in mean
Posted by Paul Levine on Thu, 22 Aug 2013 19:16:37 GMT
View Forum Message <> Reply to Message

On 2013-08-21 12:58:38 +0000, wlandsman said:

> On Tuesday, August 20, 2013 10:17:41 PM UTC-4, Paul Levine wrote:
>> On 2013-08-21 00:26:02 +0000, wlandsman said:
>>
>>
>>> Does setting !except=0 work for you? --Wayne
>>
>> Thank you for the suggestion. It does work insofar as the error>
>> message does not appear, though when I check_math I get 128, so isn't>
>> that more or less the equivalent of simply ignoring the errors?
>>
>
> Yes, setting !EXCEPT suppresses the annoying messages
>
> Why don't you want to ignore the errors (really just warnings)? It
> doesn't hurt the computer to compute floating illegal operands.
> You did say you were interested in efficiency and I'm fairly certain
> that ignoring the warning messages would be the fastest method.
> Just be sure to reset !EXCEPT afterwards so that you catch

> errors/warnings in the rest of your code. If you want to be extra
> careful then make sure the CHECK_MATH output is 128 (floating illegal
> operand) and no other math error occurred durning the mean()
> calculation.
>
> I know this doesn't answer the question you posted, which is an
> interesting problem in its own right. But sometimes the best way to
> untie a knot is to cut it with a knife. --Wayne

Thank you for the reply. The reason I was trying to avoid simply ignoring those errors was because I was afraid that doing so would mean missing other errors that were potentially less innocuous. I didn't think to reset !EXCEPT afterwards; that simple solution seems to clearly be the best one for me at this point, as I won't have to worry about missing other errors. The suggestion is much appreciated, as I now see how I can cut the knot with a knife but still be able to tie it back up again afterwards :)

Subject: Re: avoiding "floating illegal operand" errors with /nan keyword in mean
Posted by Paul Levine on Thu, 22 Aug 2013 19:26:46 GMT
View Forum Message <> Reply to Message

On 2013-08-21 21:11:45 +0000, Fabien said:

> Hi Paul,
>
> I also had the same problem:
> https://groups.google.com/forum/#!topic/comp.lang.idl-pvwave /XmPXtQE6VZ0
>
> It is not necessary to use a for loop to avoid the warnings, you just
> need to write your own MEAN() where you check for cases with
> TOTAL(FINITE(data), dimension) eq 0 but you have to decide if it worth
> it or not...

Thanks, that thread is an interesting read. I was thinking I would probably just use Wayne Landsman's suggestion of ignoring the math errors. But your suggestion of writing a new MEAN() got me looking, and I found the AVG() function in the astronomy library, which looks like it does just that, and was in fact heavily modified by Wayne Landsman himself.
