Subject: Please, please, please can we have a missing data color for image()
Posted by Gordon Farquharson on Wed, 11 Jun 2014 23:04:10 GMT
View Forum Message <> Reply to Message

The function graphics routine image() has a number of nice features, including the ability to view the value of a pixel (and coordinates) under the cursor, to zoom interactively, and to assign a colorbar with matching colors in a sensible way. However, to get all of these features, one has to let image() bytscl the data, i.e., The Function Graphics (TM) Way:

```
data = findgen(100, 100) - 5000.
im1 = image(data, RGB_TABLE=70, POSITION=[0.1, 0.1, 0.9, 0.9])
cb1 = colorbar(TARGET=im1)
```

And Bliss and Happiness ensues in IDL heaven...

(Cue scratched record sound.)

... except when the data contains missing values:

```
data = findgen(100, 100) - 5000.
data[60:69,60:69] = !values.f_nan
im1 = image(data, RGB_TABLE=70, POSITION=[0.1, 0.1, 0.9, 0.9])
cb1 = im1.colorbar(TARGET=im1)
```

Notice the hideous red square where the missing data exists. Visually, this is terrible. (My eyes are still in therapy.)

The classic way of solving this problem is to bytscl the data manually, and to reserve indices in the color table for foreground and background colors, e.g.,

```
data = findgen(100, 100) - 5000.
data[60:69,60:69] = !values.f_nan
img = bytscl(data, MIN=-5000., MAX=5000., TOP=254B) + 1B
img[where(~finite(data), /NULL)] = 0B
ct = colortable(70)
ct[0,*] = 255B * make_array(3, /BYTE, VALUE=1B)
ct[255,*] = 0B * make_array(3, /BYTE, VALUE=1B)
im1 = image(img, RGB_TABLE=ct, POSITION=[0.1, 0.1, 0.9, 0.9])
cb1 = colorbar(TARGET=im1)
```

The hideous red square of missing data has been replaced by a pleasant white (background color) square, and all is well with the world ...

... except, that we've broken most of the nice functionality of image(), because the label values in the colorbar are the img byte values :-(

Oh IDL gods, I beseech thee: Please, please, please can we have a missing data color for image(). Ideally, I'd like to have a keywords like MISSING_VALUE and MISSING_COLOR. If the

embarrassment of David having this functionality in his graphics routines is not enough to persuade the IDL gods to add this to image(), then the fact that image() is almost useless without it must convince you.

One last note: I have applied my IDL Kung fu [1] Master skills to tricking IDL into displaying a background color using the alpha channel, and this works great for bitmap formats that support transparency, but alas, it fails dismally when I try to create publication quality EPS graphics.

Gordon

[1] http://en.wikipedia.org/wiki/Kung_fu_%28term%29

Subject: Re: Please, please, please can we have a missing data color for image()
Posted by Fabzi on Thu, 12 Jun 2014 08:38:39 GMT
View Forum Message <> Reply to Message

Thanks Gordon! I completely agree

Subject: Re: Please, please, please can we have a missing data color for image()
Posted by chris_torrence@NOSPAM on Fri, 13 Jun 2014 21:48:14 GMT
View Forum Message <> Reply to Message

On Thursday, June 12, 2014 2:38:39 AM UTC-6, Fabien wrote:
> Thanks Gordon! I completely agree

Okay, here's some code to try. I didn't add the MISSING_VALUE/COLOR keywords yet. Instead, I'm just keying off of NaN values.

Open up lib/components/idlitvisimage__define.pro, and navigate to around line 1965. Modify the following block of code (I've included bits of code before & after for reference).

```
    pImageData = self->ByteScaleData(pImgData, data, nPlanes)
    ; Single-channel images must have a palette.
    ; This will also retrieve the red, green, blue values.
    if (nPlanes eq 1) then $
        self->EnsurePalette, red, green, blue

**** NEW CODE
    ; Handle float images with missing data.
    if (~self._isByteData && nPlanes eq 1) then begin
      good = FINITE(*pImgData)
      if (~ARRAY_EQUAL(good, 1b)) then begin
        nPlanes = 2
        d = TEMPORARY(data)
        data = BYTARR(imgDims[0], imgDims[1], 2)
```

```
        data[0,0,0] = TEMPORARY(d)
        data[0,0,1] = 255b*good
      endif
    endif
**** END NEW CODE

    ; Map projections. This may modify both the data and nPlanes.
    self->_UpdateMapProjection, data, nPlanes, red, green, blue
    imgDims = (SIZE(data, /DIMENSIONS))[0:1]
```

Then, try your test case, but be sure to use /BITMAP when writing to the PDF file (to avoid alpha channel issues):

```
data = findgen(100, 100) - 5000.
data[30:60,30:60] = !values.f_nan
im1 = image(data, RGB_TABLE=70, POSITION=[0.1, 0.1, 0.9, 0.9])
cb1 = colorbar(TARGET=im1)
im1.save,'test.pdf',/bitmap
```

Let me know if this works. If it does, I'll think about adding the MISSING_* keywords.

Cheers,
Chris
ExelisVIS

Subject: Re: Please, please, please can we have a missing data color for image()
Posted by Gordon Farquharson on Mon, 16 Jun 2014 19:27:28 GMT
View Forum Message <> Reply to Message

Hi Chris

Thanks very much for the test code. I've included a diff below just to make sure that I inserted it correctly.

The new code definitely works. The missing data is displayed in the background color on the screen. As you said, it produces incorrect results if you don't use the BITMAP keyword with the save method.

```
--- idlitvisimage__define.pro.orig 2014-06-16 12:04:40.000000000 -0700
+++ idlitvisimage__define.pro 2014-06-16 12:08:29.000000000 -0700
@@ -1965,6 +1965,18 @@
         self->EnsurePalette, red, green, blue


+    ; Handle float images with missing data.
+    if (~self._isByteData && nPlanes eq 1) then begin
+        good = FINITE(*pImgData)
```

```
+       if (~ARRAY_EQUAL(good, 1b)) then begin
+          nPlanes = 2
+          d = TEMPORARY(data)
+          data = BYTARR(imgDims[0], imgDims[1], 2)
+          data[0,0,0] = TEMPORARY(d)
+          data[0,0,1] = 255b*good
+       endif
+    endif
+
    ; Map projections. This may modify both the data and nPlanes.
    self->_UpdateMapProjection, data, nPlanes, red, green, blue
    imgDims = (SIZE(data, /DIMENSIONS))[0:1]
```

Gordon

On Friday, June 13, 2014 2:48:14 PM UTC-7, Chris Torrence wrote:
> On Thursday, June 12, 2014 2:38:39 AM UTC-6, Fabien wrote:
>
>> Thanks Gordon! I completely agree
>
>
>
> Okay, here's some code to try. I didn't add the MISSING_VALUE/COLOR keywords yet. Instead, I'm just keying off of NaN values.
>
>
>
> Open up lib/components/idlitvisimage__define.pro, and navigate to around line 1965. Modify the following block of code (I've included bits of code before & after for reference).
>
>
>
```
>     pImageData = self->ByteScaleData(pImgData, data, nPlanes)
>
>     ; Single-channel images must have a palette.
>
>     ; This will also retrieve the red, green, blue values.
>
>     if (nPlanes eq 1) then $
>
>        self->EnsurePalette, red, green, blue
>
>
>
> **** NEW CODE
>
>     ; Handle float images with missing data.
```

```
>
>     if (~self._isByteData && nPlanes eq 1) then begin
>
>       good = FINITE(*pImgData)
>
>       if (~ARRAY_EQUAL(good, 1b)) then begin
>
>         nPlanes = 2
>
>         d = TEMPORARY(data)
>
>         data = BYTARR(imgDims[0], imgDims[1], 2)
>
>         data[0,0,0] = TEMPORARY(d)
>
>         data[0,0,1] = 255b*good
>
>       endif
>
>     endif
>
>  **** END NEW CODE
>
>
>
>     ; Map projections. This may modify both the data and nPlanes.
>
>     self->_UpdateMapProjection, data, nPlanes, red, green, blue
>
>     imgDims = (SIZE(data, /DIMENSIONS))[0:1]
>
>
>
>  Then, try your test case, but be sure to use /BITMAP when writing to the PDF file (to avoid
alpha channel issues):
>
>
>
>  data = findgen(100, 100) - 5000.
>
>  data[30:60,30:60] = !values.f_nan
>
>  im1 = image(data, RGB_TABLE=70, POSITION=[0.1, 0.1, 0.9, 0.9])
>
>  cb1 = colorbar(TARGET=im1)
>
>  im1.save,'test.pdf',/bitmap
>
```

>
>
> Let me know if this works. If it does, I'll think about adding the MISSING_* keywords.
>
>
>
> Cheers,
>
> Chris
>
> ExelisVIS
