Subject: STRETCH: A bug?
Posted by timrobishaw on Fri, 19 Mar 2004 10:20:19 GMT
View Forum Message <> Reply to Message

Hey there,

So I've been thinking too much about color tables. I took a look at IDL's STRETCH just to make sure it does as it advertises, but I'm mostly sure it doesn't. I'm not so sure I want to run around saying it's a bug, so I thought I'd run it by you guys for opinions.

The short of it, for those who might not want to read the junk below is that I'm really convinced that STRETCH should be using BYTSCL() rather than LONG() to determine the color table index mapping.

Anyway, here's the deal, in lots of (hopefully) easy-to-follow detail:

You have a color table running from 0 to NC-1, where NC is the number of colors in the color table, !D.TABLE_SIZE. You would now like to stretch the entire original color table over the indices LOW to HIGH in the new color table and set all the indices below LOW to the value 0 and then set all the indices above high to (NC-1). This stretched color table will also have NC indices and will replace the original color table.

For instructional purposes, let's pretend we're running a few Netscapes and we have 18 available colors:

```
IDL> NC = 18
IDL> LOW = 6
IDL> HIGH = 14
```

IDL's STRETCH achieves this stretching by first finding the slope of the new color table:

```
IDL> slope = float(nc-1)/(high-low)
```

Then it finds the intercept, i.e., the index of the original color table that would occupy the very first index of our new color table after the stretching is done.

```
IDL> intercept = -slope*low
```

RSI then transforms to the new fractional color table indices:

```
IDL> pf = findgen(nc)*slope+intercept
```

So you're scaling your original color table such that you're guaranteed that new color table's value of ZERO will be located at index LOW and the value of (NC-1) will be found at index HIGH:

```
IDL> print, pf
    -12.7500     -10.6250     -8.50000     -6.37500     -4.25000
    -2.12500      0.00000      2.12500      4.25000      6.37500
     8.50000      10.6250      12.7500      14.8750      17.0000
     19.1250      21.2500      23.3750
IDL> print, where(pf eq 0), where(pf eq nc-1)
           6
          14
```

Now, what we'd really like at this step is to evenly divvy up the fractional indices between LOW and HIGH. Unfortunately, here is where I think RSI makes a big mistake! They decided (in 1983) to operate on the fractional values using the LONG() function:

```
IDL> print, long(pf)
         -12          -10           -8           -6           -4
          -2            0            2            4            6
           8           10           12           14    17
          19           21           23
```

What this does is subtle, yet important! LONG() will always return the *FLOOR()* of the fractional value! (OK, it will be the FLOOR() if the fractional value is POSITIVE, but we'll only be interested in the positive indices!) So, the LONG()ing of the fractional values creates the undesirable situation that there will be only one, single value of the index (NC-1) no matter what the slope! If you think about this in terms of making a histogram (which is really what you are doing here!) you're reserving an entire bin for the single value of (PF EQ NC-1). Just to make that clear, make a histogram!

```
IDL> h = histogram(pf,min=0,max=nc-1,binsize=1.0)
IDL> print, byte(where(h gt 0))
   0   2   4   6   8  10  12  14  17
```

OK, you get the same result as the LONG()ing, but, again, what you've done is put the left edge of your last bin on 17 such that you're wasting a whole bin! Even worse, you've made the bins a little larger than you should have (BINSIZE=1.0), so you've affected the entire distribution of indices.

What you'd really like is to byte-scale the values between LOW and HIGH in the original color table:

```
IDL> print, bytscl(pf,min=0,max=nc-1,top=nc-1)
   0   0   0   0   0   0   0   2   4   6   8  11  13  15  17  17  17
```

17

This is definitely different than simply LONG()ing the fractional
indices. How do we understand this in terms of a histogram? Well,
now we're making NC bins stretched over the range 0 to (NC-1), so the
bin size should be (NC-1)/NC rather than 1.0! So making just such a
histogram exactly reproduces the BYTSCL result:

```
IDL> h =  histogram(pf,min=0,max=nc-1,binsize=float(nc-1)/nc,reverse=r )
IDL> h = byte(where(h gt 0,nh))
IDL> print, h
   0   2   4   6   9  11  13  15  18
IDL> b = byte(pf < (nc-1) > 0)
IDL> for i = 0, nh-1 do b[R[R[h[i]]:R[h[i]+1]-1]] = (h[i]<(nc-1))
IDL> print, b
   0   0   0   0   0   0   0   2   4   6   9  11  13  15  17  17  17
17
```

So, IDL got most of STRETCH right, but using LONG() instead of
BYTSCL() was a bad choice.

I guess the easy way to do the color table index mapping would be like
so:

```
ctindx = bytscl(((findgen(nc)-low)/(high-low) < 1.0 > 0.0)^gamma,
MIN=0, MAX=1)
```

So, RSI's algorithm is a little flawed.
Isn't this a "bug"... if you, ya know, care about doing things right?

-Tim.