Customising matplotlib cmaps

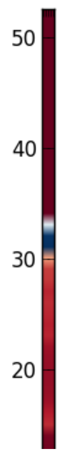
I have some normalised histogram data in array of shape (12,1):

```
>>> hnorm

   array([[ 0.        ],
       [ 0.        ],
       [ 0.01183432],
       [ 0.0295858 ],
       [ 0.04142012],
       [ 0.04142012],
       [ 0.03550296],
       [ 0.01775148],
       [ 1.        ],
       [ 0.98816568],
       [ 0.56213018],
       [ 0.        ]])
```

I'd like to plot this in 'heatmap' style. I am doing this like so:

```
import matplotlib.pyplot as plt
plt.imshow(hnorm, cmap='RdBu',origin='lower')
```

This works (axis formatting aside).



However, I'd like to customise the colormap to fade from white to Red. I have attempted:

```
import matplotlib.colors as col

cdict = {'red': ((0.00, 0.07, 0.14),
        (0.21, 0.28, 0.35),
        (0.42, 0.49, 0.56),
        (0.63, 0.70, 0.77),
        (0.84, 0.91, 0.99)),
        'green': ((0.0, 0.0, 0.0),
        (0.0, 0.0, 0.0),
        (0.0, 0.0, 0.0),
        (0.0, 0.0, 0.0),
        (0.0, 0.0, 0.0)),
        'blue': ((0.0, 0.0, 0.0),
        (0.0, 0.0, 0.0),
        (0.0, 0.0, 0.0),
        (0.0, 0.0, 0.0),
        (0.0, 0.0, 0.0))}
cmap1 = col.LinearSegmentedColormap('my_colormap',cdict,N=256,gamma=0.75)
plt.imshow(hnorm, cmap=cmap1,origin='lower')
```

This fails. Any ideas what I am doing wrong?

python numpy matplotlib plot

asked Sep 6 '13 at 14:57
atomh33ls
2,007 5 23

2 If all you really want is to have a white-to-red colormap, you can just use `cmap='Reds'` (or `cmap='Reds_r'` for the opposite direction). – askewchan Sep 6 '13 at 15:00

add comment

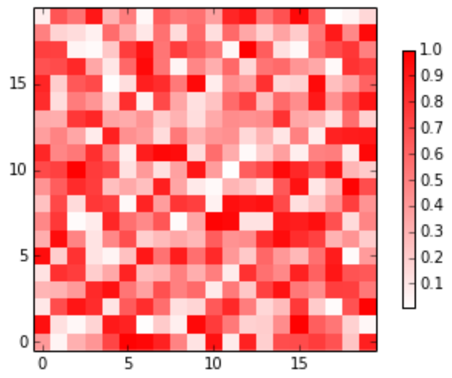
1 Answer

The cmap 'Reds' as askewchan is suggesting is simpler and (imo) also better looking. But i'll answer just to show how your approach of building a custom cmap could also work.

In your color dict you have 5 entries at which you specify the color. Since you want to only use red and white you need only two enties. For white, all colors must be used which is specified by color values of 1.0 at position 0.0. For red only red should be used at position 1.0.

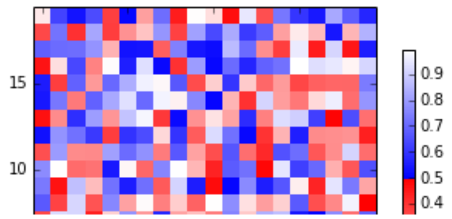
You also only provide values (other than 0) for your red tuple. This will only give you different shades of red between 'full' red and black (since you always have green and blue as 0).

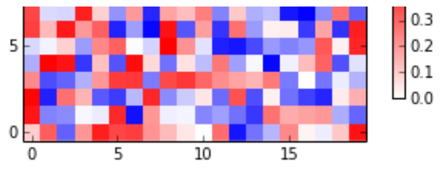
```
cdict = {'red': ((0.0, 1.0, 1.0),
                 (1.0, 1.0, 1.0)),

        'green': ((0.0, 1.0, 1.0),
                  (1.0, 0.0, 0.0)),

        'blue': ((0.0, 1.0, 1.0),
                 (1.0, 0.0, 0.0))}

my_cmap = mpl.colors.LinearSegmentedColormap('my_colormap', cdict)

plt.imshow(np.random.rand(20,20), cmap=my_cmap, origin='lower', interpolation=
plt.colorbar(shrink=.75)
```



Another example showing how the two color items allow 'jumps' in the cmap:

```
cdict = {'red': ((0.0, 1.0, 1.0), # full red
                 (0.5, 1.0, 0.0), # full red till, no red after
                 (1.0, 1.0, 1.0)), # full red

        'green': ((0.0, 1.0, 1.0), # full green
                  (0.5, 0.0, 0.0), # no green
                  (1.0, 1.0, 1.0)), # full green

        'blue': ((0.0, 1.0, 1.0), # full blue
                 (0.5, 0.0, 1.0), # no blue till, full blue after
                 (1.0, 1.0, 1.0))} # full blue

my_cmap = mpl.colors.LinearSegmentedColormap('my_colormap', cdict)

plt.imshow(np.random.rand(20,20), cmap=my_cmap, origin='lower', interpolation=
plt.colorbar(shrink=.75)
```





answered Sep 6 '13 at 15:33

Rutger Kassies
5,886 4 17

add comment

Not the answer you're looking for? Browse other questions tagged python numpy matplotlib plot or ask your own question.