To Nicole (3/8/23)

various color options:

https://matplotlib.org/stable/gallery/color/named_colors.html

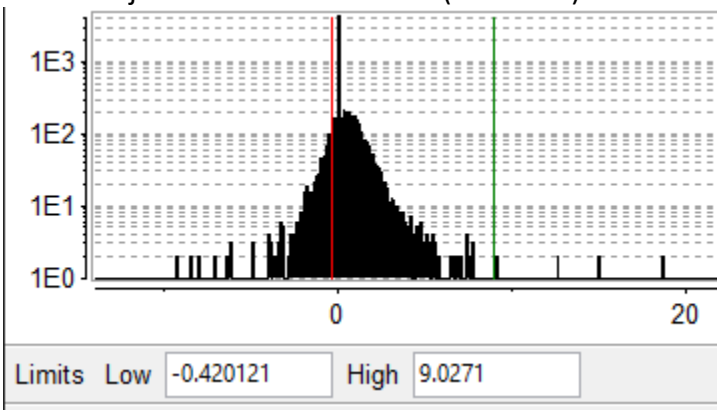
https://petercbsmith.github.io/color-tutorial.html -
> https://petercbsmith.github.io/blog_assets/colors_tutorial/plot4.png (a more complete list from an XKCD survey, which you can access by saying color='xkcd:...', with ... as a color from that .png file)

To Sam (3/10/23)

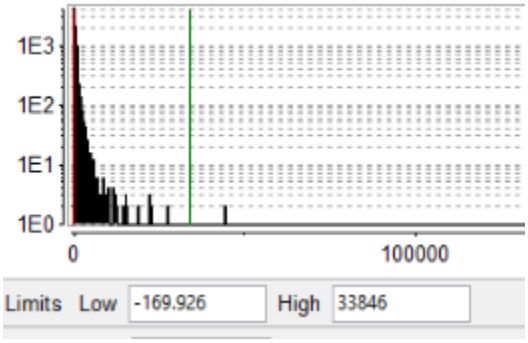
Hey Sam,

I'm still working out the matplotlib version of this, but I did some quick work on this in DS9, which I'm emailing to keep it reproducable. You might've already figured this stuff out, but hope not useless.

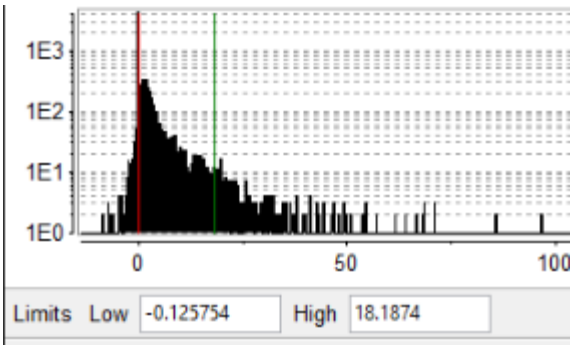
I made the slices as...
red = the jet or hot ionic material (labeled b). I scaled this as Linear and 99%+. Or with limits:



green = the continuum like you did (r). I scaled this as **Log** 99%+ (subject to change, but important thing is log puts the dimmer parts on the same scale on the much brighter inner parts...definitely misleading in terms of brightness scale, but purely to demonstrate structure of features, hopefully helps). Limits:

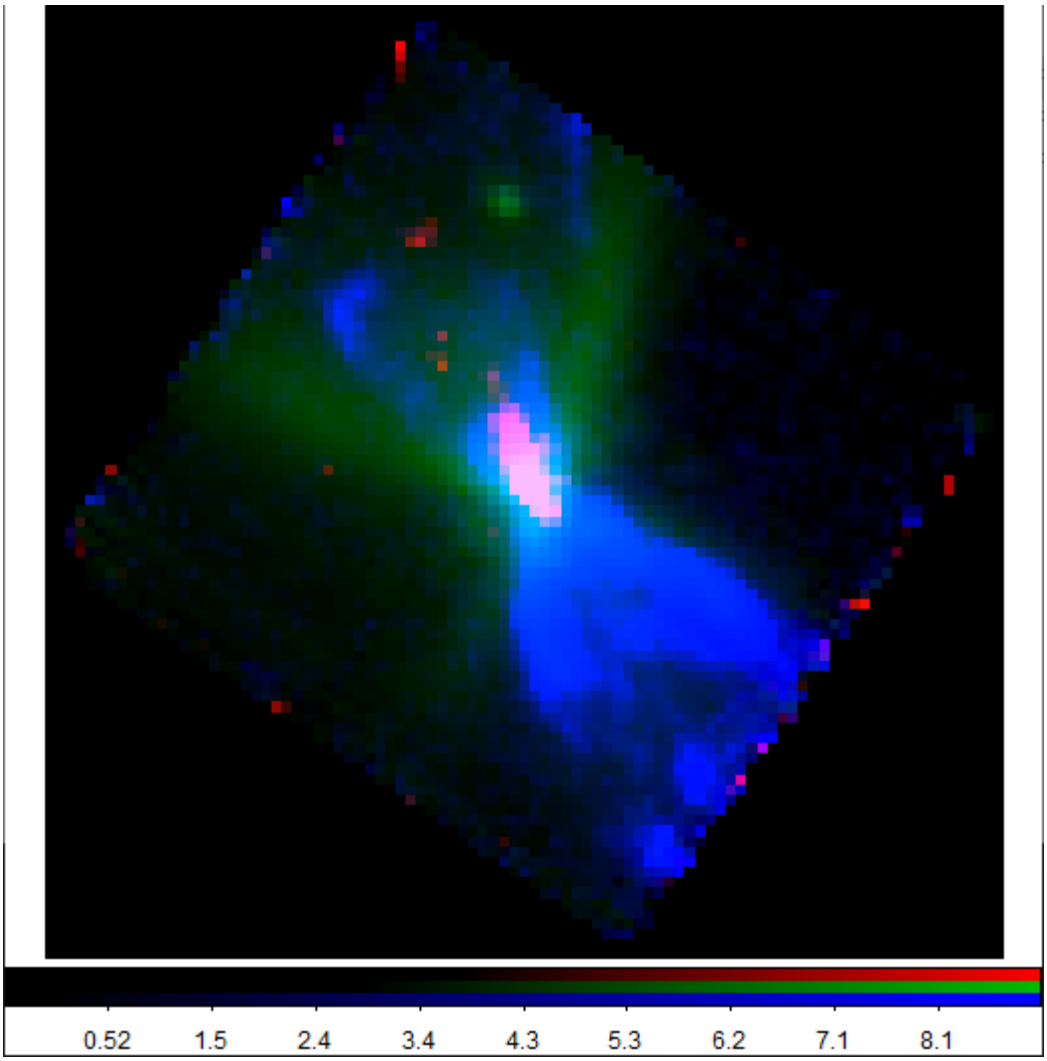


blue = the h2 molecular material (g). I scaled this as Linear ~90%. Limits:



You'll notice a general pattern that I try to put the lower limit ~0, while the upper limit is somewhere in the upper half of the histogram of values.

Here is the result. The most important thing is I made the red the jet instead of blue because the blue is so dark. We need the red as bright and distinct from the green. So the green has to be a dimmer or extended feature. The blue can be closer to the red but still has to be more extended to be seen (I still have trouble scaling it right). I think the rule of thumb will be the red and the green don't overlap where features are important, and the blue is in between (the color blending would get tricky then: green + red = yellow/brown vs red + blue = purple/magenta). I think the biggest issue is really that this blue is so dark, needs scaling to be brighter or *something*:



The gist for coding is I can probably mod either matplotlib or aplpy, but it'd be tricky and take me into next week.

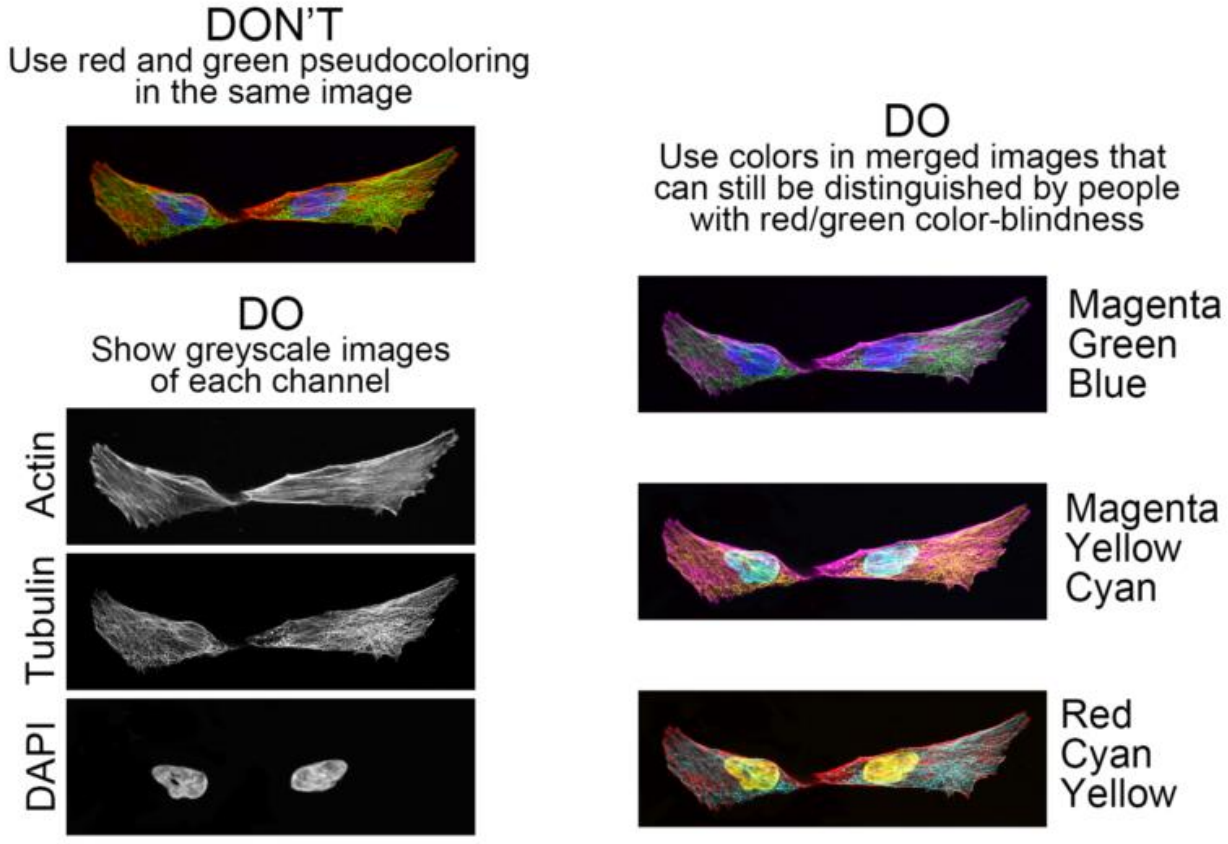
The whole idea of overlays like this comes from "color blending". For computers and light-based displays, many standards unfortunately mandate RGB-based schemes and use an additive scheme to produce mixes (traditionally, this makes "sense", you **could** use something like RYB, but that'd be awful...but maybe there was another way...). This means color blending based on non-RGB (e.g. RYB) is extremely hard. The only alternative I can think of is something like ax.imshow(data, color=colormap, alpha=alpha), where you can control color maps (see cell bio pics at end of email) and saturate layers (with alpha).

But there is one way to mod mixing methods. To change base colors, we'd convert from RGB system to another system called HSV or Hue, Saturation, and Value (aka brightness). Hue let's you control color, but the blending can become unruly (this problem is called "uniform perception", which is the issue with the dark blue and bright green here - you have to be good at colors, know someone who is, or survey friends like I do). That said, it

would give more options (e.g. https://stackoverflow.com/questions/73956440/python-to-change-image-into-a-color-that-is-not-one-of-the-images-dominant-colo or https://stackoverflow.com/questions/61184160/change-image-color-in-pil-module...or do it by command line?? https://stackoverflow.com/questions/51520369/pil-change-color-channel-intensity/51543824#51543824).

Note this all only matters if you actually want color blending, otherwise you can just pick color maps where max=color and min=black (I suspect this can be done with scaling and/or inverting the color bar), and then adjust scaling and alpha as you wish: https://matplotlib.org/stable/gallery/images_contours_and_fields/layer_images.html#sphx-glr-gallery-images-contours-and-fields-layer-images-py. If aplpy interacts with matplotlib, then it is no big deal. If not and you need how to do the projection, I can dig up how I did that from H361-C for ya.

For more on tips for coloration and ways to present 3D data, can see https://www.biorxiv.org/content/10.1101/2020.10.08.327718v2.full.pdf or a summary from the american cell bio site about this: https://www.ascb.org/wp-content/uploads/2019/09/Figure-4-accessible-IF-768x550.png



The only other tip I found indirectly from those links is you can use Principal Component Analysis to note bright features. Good for automation, but it loses physical meaning. But if the point of figures like this is just to spot nice features, it could help suggest about the scaling or how to best emphasize features. EX: https://www.askpython.com/python/examples/principal-component-analysis-for-image-data. Anyway, I probably won't get around to it, but thought I'd pass the idea off.

Reply

Forward

FA