# Seaborn Workshop

## Sequential color palettes

- The second major class of color palettes is called sequential.
- This kind of color mapping is appropriate when data range from relatively low or unintersting values to relatively high or interesting values.
- Although there are cases where you will want discrete colors in a sequential palette, its more common to use them as a colormap in functions like `kdeplot()` or `corrplot()` (along with similar matplotlib functions).

## Seaborn Workshop

- Its common to see colormaps like jet (or other rainbow palettes) used in this case, becuase the range of hues gives the impression of providing additional information about the data.
- However, colormaps with large hue shifts tend to introduce discontinuities that dont exist in the data, and our visual system isnt able to naturally map the rainbow to quantitative distinctions like high or low.

## Seaborn Workshop

- The result is that these visualizations end up being more like a puzzle, and they obscure patterns in the data rather than revealing them.
- The jet palette is particularly bad because the brightest colors, yellow and cyan, are used for intermediate data values. This has the effect of emphasizing uninteresting (and arbitrary) values while demphasizing the extremes.

## Seaborn Workshop

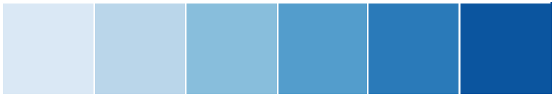
- For sequential data, its better to use palettes that have at most a relatively subtle shift in hue accompanied by a large shift in brightness and saturation.
- This approach will naturally draw the eye to the relatively important parts of the data.

## Seaborn Workshop

The Color Brewer library has a great set of these palettes. Theyre named after the dominant color (or colors) in the palette.

## Seaborn Workshop

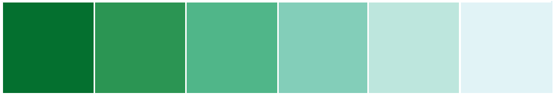
```
sns.palplot(sns.color_palette("Blues"))
```



## Seaborn Workshop

Like in matplotlib, if you want the lightness ramp to be reversed, you can add a _r suffix to the palette name.

```
sns.palplot(sns.color_palette("BuGn_r"))
```

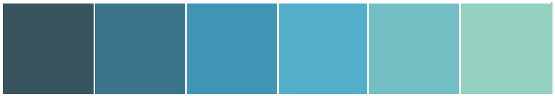


## Seaborn Workshop

- Seaborn also adds a trick that allows you to create dark palettes, which do not have as wide a dynamic range.
- This can be useful if you want to map lines or points sequentially, as brighter-colored lines might otherwise be hard to distinguish.

## Seaborn Workshop

```
sns.palplot(sns.color_palette("GnBu_d"))
```



## Seaborn Workshop

- Remember that you may want to use the `choose_colorbrewer_palette()` function to play with the various options, and you can set the `as_cmap` argument to `True` if you want the return value to be a colormap object that you can pass to seaborn or matplotlib functions.

# Seaborn Workshop

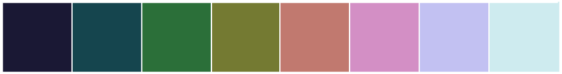
**Sequential palettes with** `cubehelix_palette()`

- The cubehelix color palette system makes sequential palettes with a linear increase or decrease in brightness and some variation in hue.
- This means that the information in your colormap will be preserved when converted to black and white (for printing) or when viewed by a colorblind individual.

## Seaborn Workshop

Matplotlib has the default cubehelix version built into it:

```
sns.palplot(sns.color_palette("cubehelix", 8))
```



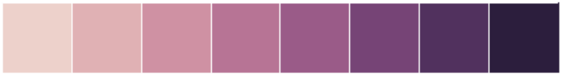
Seaborn adds an interface to the cubehelix system so that you can make a variety of palettes that all have a well-behaved linear brightness ramp.

## Seaborn Workshop

The default palette returned by the seaborn `cubehelix_palette()` function is a bit different from the matplotlib default in that it does not rotate as far around the hue wheel or cover as wide a range of intensities. It also reverses the order so that more important values are darker:

## Seaborn Workshop

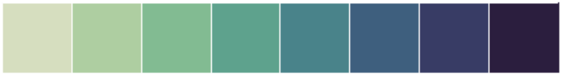
```
sns.palplot(sns.cubehelix_palette(8))
```



## Seaborn Workshop

Other arguments to cubehelix_palette() control how the palette looks. The two main things youll change are the start (a value between 0 and 3) and rot, or number of rotations (an arbitrary value, but probably within -1 and 1),

```
sns.palplot(sns.cubehelix_palette(8, start=.5, rot=-.75
```

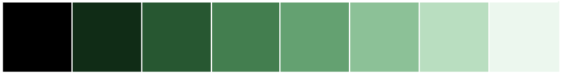


You can also control how dark and light the endpoints are and even reverse the ramp:

## Seaborn Workshop

By default you

```
sns.palplot(sns.cubehelix_palette(8, start=2, rot=0, da
```

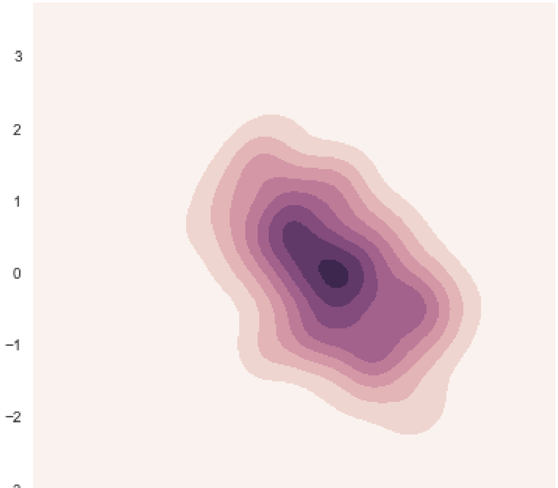


## Seaborn Workshop

By default you just get a list of colors, like any other seaborn palette, but you can also return the palette as a colormap object that can be passed to seaborn or matplotlib functions using `as_cmap=True`.

## Seaborn Workshop

```
x, y = np.random.multivariate_normal([0, 0], [[1, -.5],
cmap = sns.cubehelix_palette(light=1, as_cmap=True)
sns.kdeplot(x, y, cmap=cmap, shade=True);
```



## Seaborn Workshop

- To help select good palettes or colormaps using this system, you can use the `choose_cubehelix_palette()` function in a notebook to launch an interactive app that will let you play with the different parameters.
- Pass `as_cmap=True` if you want the function to return a colormap (rather than a list) for use in function like hexbin.

## Seaborn Workshop

- For a simpler interface to custom sequential palettes, you can use `light_palette()` or `dark_palette()`, which are both seeded with a single color and produce a palette that ramps either from light or dark desaturated values to that color.
- These functions are also accompanied by the `choose_light_palette()` and `choose_dark_palette()` functions that launch interactive widgets to create these palettes.

## Seaborn Workshop

```
sns.palplot(sns.light_palette("green"))
```

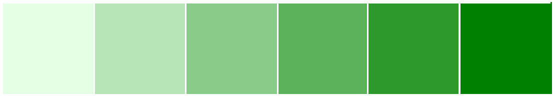


Figure:

## Seaborn Workshop

```
sns.palplot(sns.dark_palette("purple"))
```

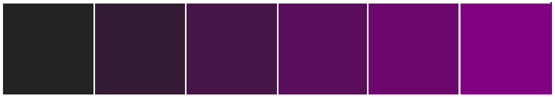


Figure:

## Seaborn Workshop

These palettes can also be reversed.

```
sns.palplot(sns.light_palette("navy", reverse=True))
```

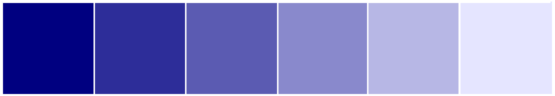
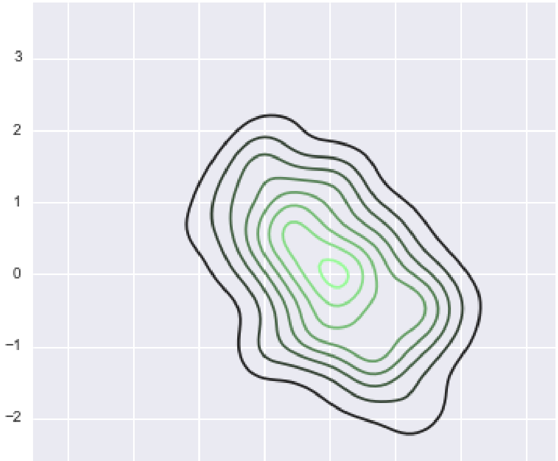


Figure:

## Seaborn Workshop

They can also be used to create colormap objects rather than lists of colors.

```
pal = sns.dark_palette("palegreen", as_cmap=True)
sns.kdeplot(x, y, cmap=pal);
```



## Seaborn Workshop

- By default, the input can be any valid matplotlib color.
- Alternate interpretations are controlled by the input argument.
- Currently you can provide tuples in hls or husl space along with the default rgb, and you can also seed the palette with any valid xkcd color.

## Seaborn Workshop

```
sns.palplot(sns.light_palette((210, 90, 60), input="hus
```

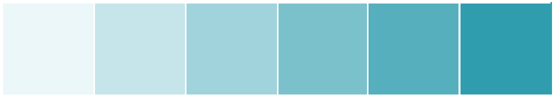
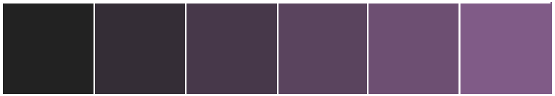


Figure:

## Seaborn Workshop

```
sns.palplot(sns.dark_palette("muted purple", input="xkc
```



Note that the default input space for the interactive palette widgets is husl, which is different from the default for the function itself, but much more useful in this context.