Choosing color palettes

- Color is more important than other aspects of figure style because color can reveal patterns in the data if used effectively or hide those patterns if used poorly.
- There are a number of great resources to learn about good techniques for using color in visualizations, I am partial to this series of blog posts from Rob Simmon and this more technical paper.
- The matplotlib docs also now have a nice tutorial that illustrates some of the perceptual properties of the built in colormaps.

Seaborn Workshop

Seaborn makes it easy to select and use color palettes that are suited to the kind of data you are working with and the goals you have in visualizing it.

```
%matplotlib inline
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(rc={"figure.figsize": (6, 6)})
np.random.seed(sum(map(ord, "palettes")))
```

Building color palettes with `color_palette()` The most important function for working with discrete color palettes is `color_palette()`. This function provides an interface to many (though not all) of the possible ways you can generate colors in seaborn, and its used internally by any function that has a palette argument (and in some cases for a color argument when multiple colors are needed).

Seaborn Workshop

- `color_palette()` will accept the name of any seaborn palette or matplotlib colormap (except jet, which you should never use).
- It can also take a list of colors specified in any valid matplotlib format (RGB tuples, hex color codes, or HTML color names).
- The return value is always a list of RGB tuples.

Seaborn Workshop

Finally, calling `color_palette()` with no arguments will return the current default color cycle.
A corresponding function, set_palette(), takes the same arguments and will set the default color cycle for all plots. You can also use `color_palette()` in a with statement to temporarily change the default palette (see below).

Seaborn Workshop

- It is generally not possible to know what kind of color palette or colormap is best for a set of data without knowing about the characteristics of the data.
- Following that, well break up the different ways to use `color_palette()` and other seaborn palette functions by the three general kinds of color palettes: qualitative, sequential, and diverging.

Seaborn Workshop

Qualitative color palettes Qualitative (or categorical) palettes are best when you want to distinguish discrete chunks of data that do not have an inherent ordering.
When importing seaborn, the default color cycle is changed to a set of six colors that evoke the standard matplotlib color cycle while aiming to be a bit more pleasing to look at.

Seaborn Workshop

```
current_palette = sns.color_palette()
sns.palplot(current_palette)
```

There are six variations of the default theme, called deep, muted, pastel, bright, dark, and colorblind.

Using circular color systems

- When you have more than six categories to distinguish, the easiest thing is to draw evenly-spaced colors in a circular color space (such that the hue changes which keeping the brightness and saturation constant).
- This is what most seaborn functions default to when they need to use more colors than are currently set in the default color cycle.

The most common way to do this uses the hls color space, which is a simple transformation of RGB values.

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

Figure:

Seaborn Workshop

There is also the `hls_palette()` function that lets you control the lightness and saturation of the colors.

```
sns.palplot(sns.hls_palette(8, l=.3, s=.8))
```

Seaborn Workshop

- However, because of the way the human visual system works, colors that are even intensity in terms of their RGB levels wont necessarily look equally intense.
- We perceive yellows and greens as relatively brighter and blues as relatively darker, which can be a problem when aiming for uniformity with the hls system.