Diverging color palettes

- The third class of color palettes is called diverging. These are used for data where both large low and high values are interesting.
- There is also usually a well-defined midpoint in the data.
- For instance, if you are plotting changes in temperature from some baseline timepoint, it is best to use a diverging colormap to show areas with relative decreases and areas with relative increases.

Seaborn Workshop

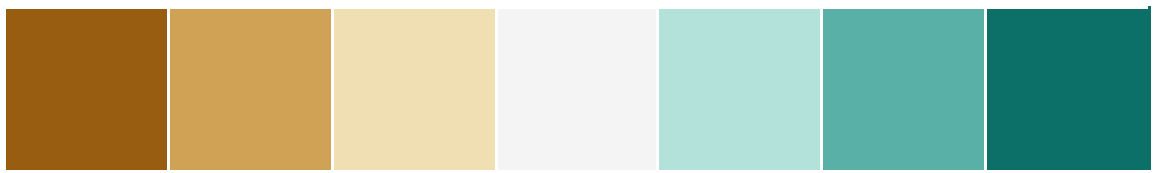
- The rules for choosing good diverging palettes are similar to good sequential palettes, except now you want to have two relatively subtle hue shifts from distinct starting hues that meet in an under-emphasized color at the midpoint.
- Its also important that the starting values are of similar brightness and saturation.

Seaborn Workshop

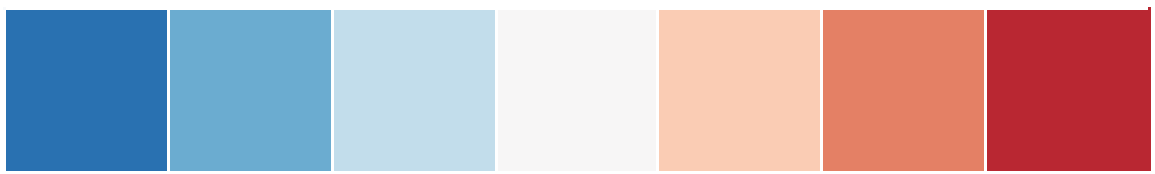
- Its also important to emphasize here that using red and green should be avoided, as a substantial population of potential viewers will be unable to distinguish them.
- It should not surprise you that the **Color Brewer** library comes with a set of well-choosen diverging colormaps.

Seaborn Workshop

```
sns.palplot(sns.color_palette("BrBG", 7))
```



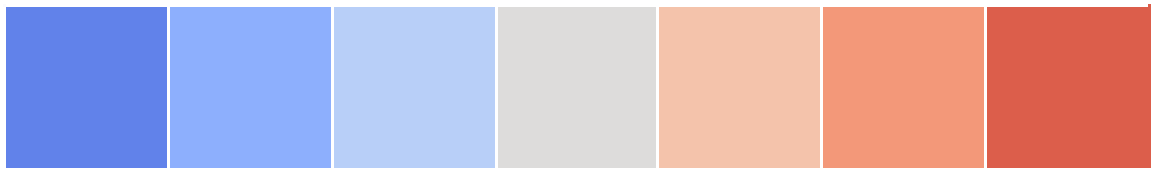
```
sns.palplot(sns.color_palette("RdBu_r", 7))
```



Another good choice that is built into matplotlib is the coolwarm palette. Note that this colormap has less contrast between the middle values and the extremes.

Seaborn Workshop

```
sns.palplot(sns.color_palette("coolwarm", 7))
```

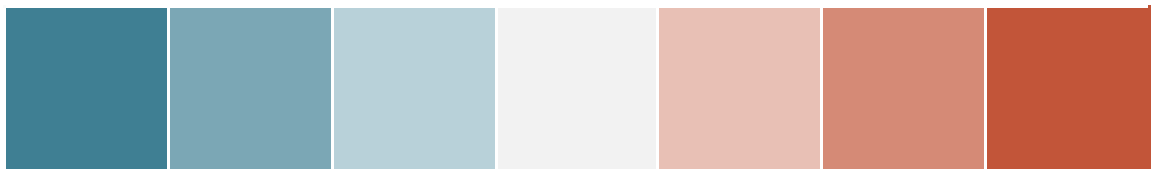


Seaborn Workshop

- You can also use the seaborn function `diverging_palette()` to create a custom colormap for diverging data. (Naturally there is also a companion interactive widget, `choose_diverging_palette()`). This function makes diverging palettes using the husl color system.
- You pass it two hues (in degreees) and, optionally, the lightness and saturation values for the extremes.
- Using husl means that the extreme values, and the resulting ramps to the midpoint, will be well-balanced

Seaborn Workshop

```
sns.palplot(sns.diverging_palette(220, 20, n=7))
```



```
sns.palplot(sns.diverging_palette(145, 280,
            s=85, l=25, n=7))
```

Seaborn Workshop

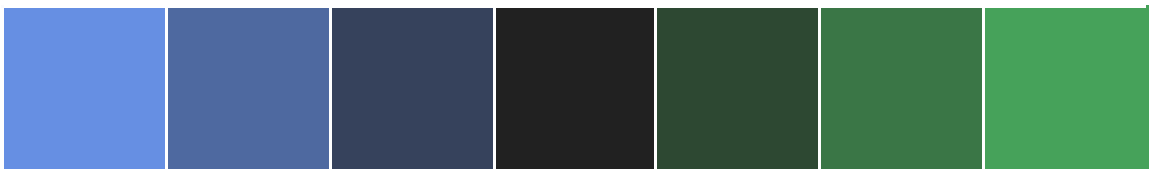
The sep argument controls the width of the separation between the two ramps in the middle region of the palette.

```
sns.palplot(sns.diverging_palette(10, 220, sep=80, n=7)
```

Seaborn Workshop

Its also possible to make a palette with the midpoint is dark rather than light.

```
sns.palplot(sns.diverging_palette(255, 133,
     l=60, n=7, center="dark"))
```



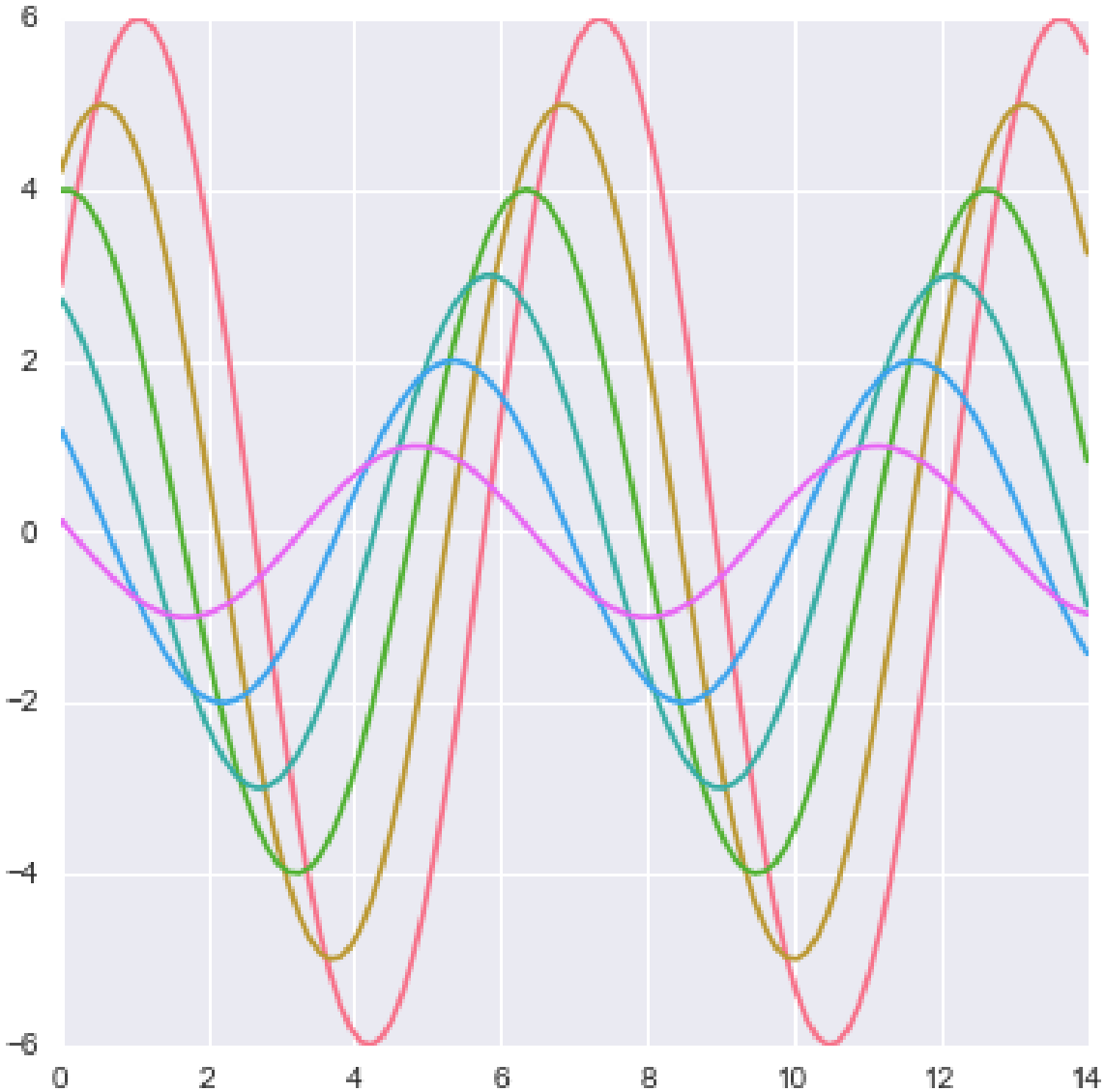
Seaborn Workshop

- The `color_palette()` function has a companion called `set_palette()`.
- The relationship between them is similar to the pairs covered in the aesthetics tutorial.
- `set_palette()` accepts the same arguments as `color_palette()`, but it changes the default matplotlib parameters so that the palette is used for all plots.

Seaborn Workshop

```
def sinplot(flip=1):
    x = np.linspace(0, 14, 100)
    for i in range(1, 7):
        y = np.sin(x + i * .5) * (7 - i)
        plt.plot(x, y * flip)
        sns.set_palette("husl")
        sinplot()
```

Seaborn Workshop



Seaborn Workshop

- The `color_palette()` function can also be used in a with statement to temporarily change the color palette.

```
with sns.color_palette("PuBuGn_d"):
sinplot()
```