IMPROVING YOUR DATA VISUALIZATIONS IN PYTHON

Color in visualizations

Nick Strayer
Instructor

How color is used

- Differentiates classes of data
- Encodes continuous values
- Should be used *carefully*

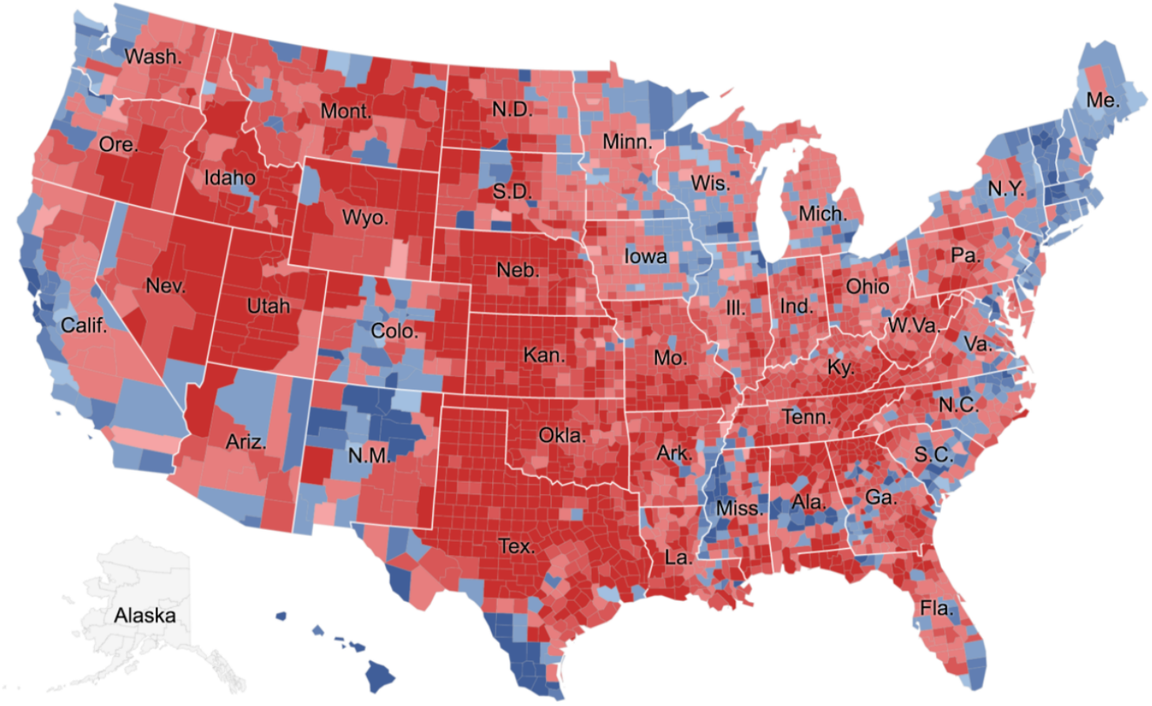
Color can be beautiful

- Boring → eye-catching
- Variety is good

 start greyscale map compared to colorful blue map
https://assets.datacamp.com/production/repositories/3841/datas

Color can be polarizing

- Meaning is often applied to colors via culture/ personal experience



Source: NYT

Color preferences can vary

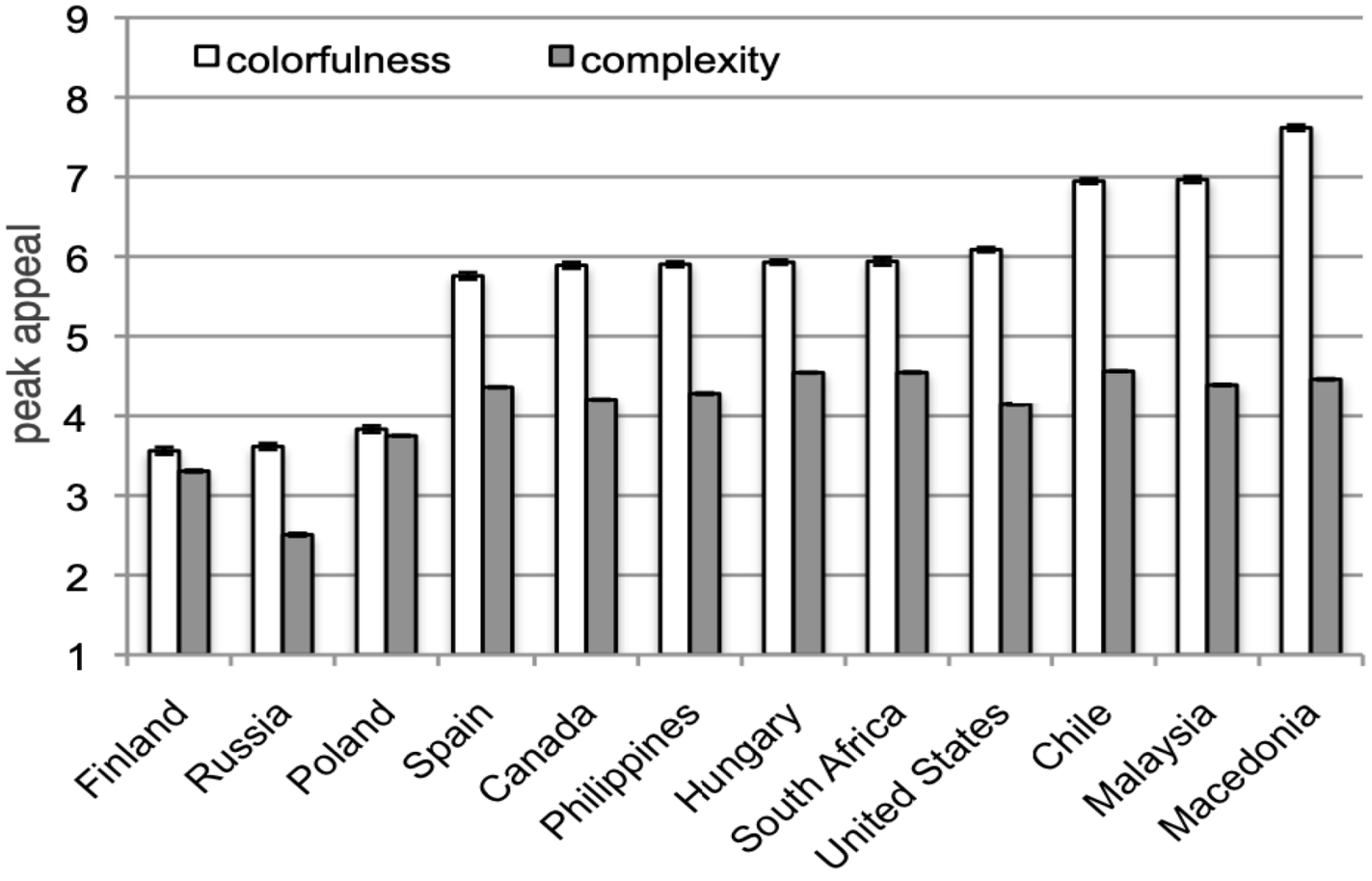


Figure by Katharina Reinecke & Krzysztof Z. Gajos, 2014

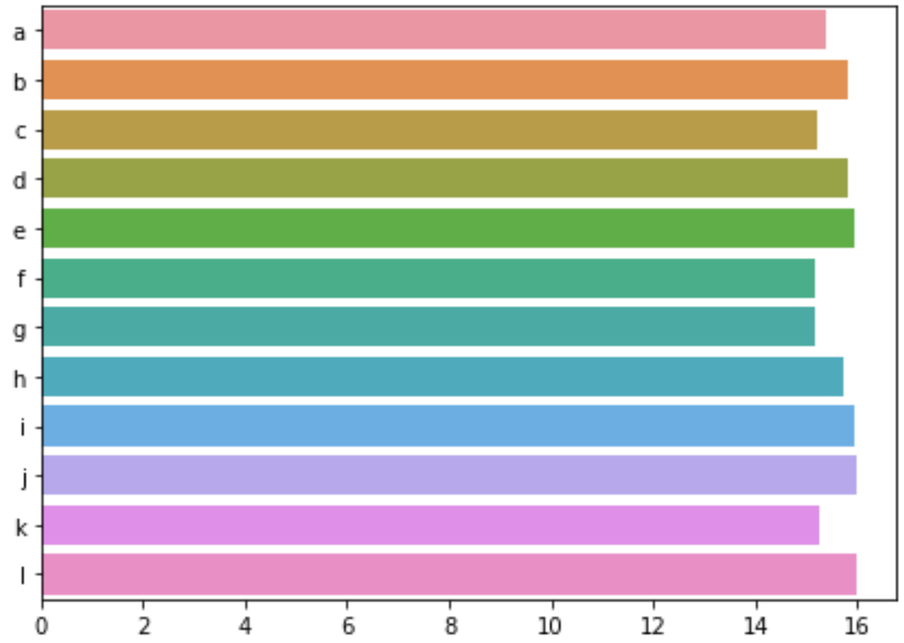
Color can be misleading...

"It is evident that the color-size illusion is present in a marked degree [no matter what] arrangement."

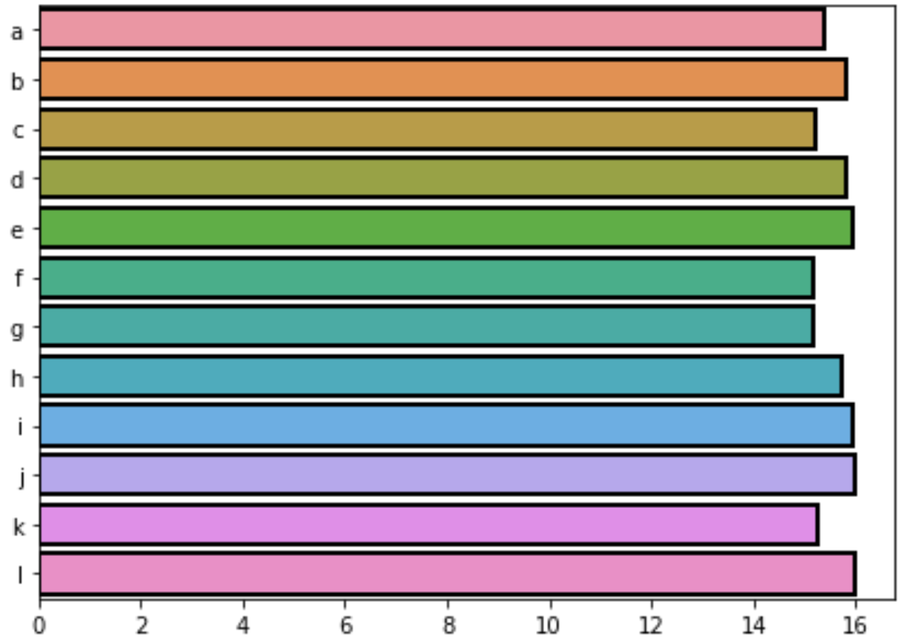
C.J. Warden & E.L. Flynn, 1926

A remedy for the color-size illusion

```
sns.barplot(x = values, y = ids)
```

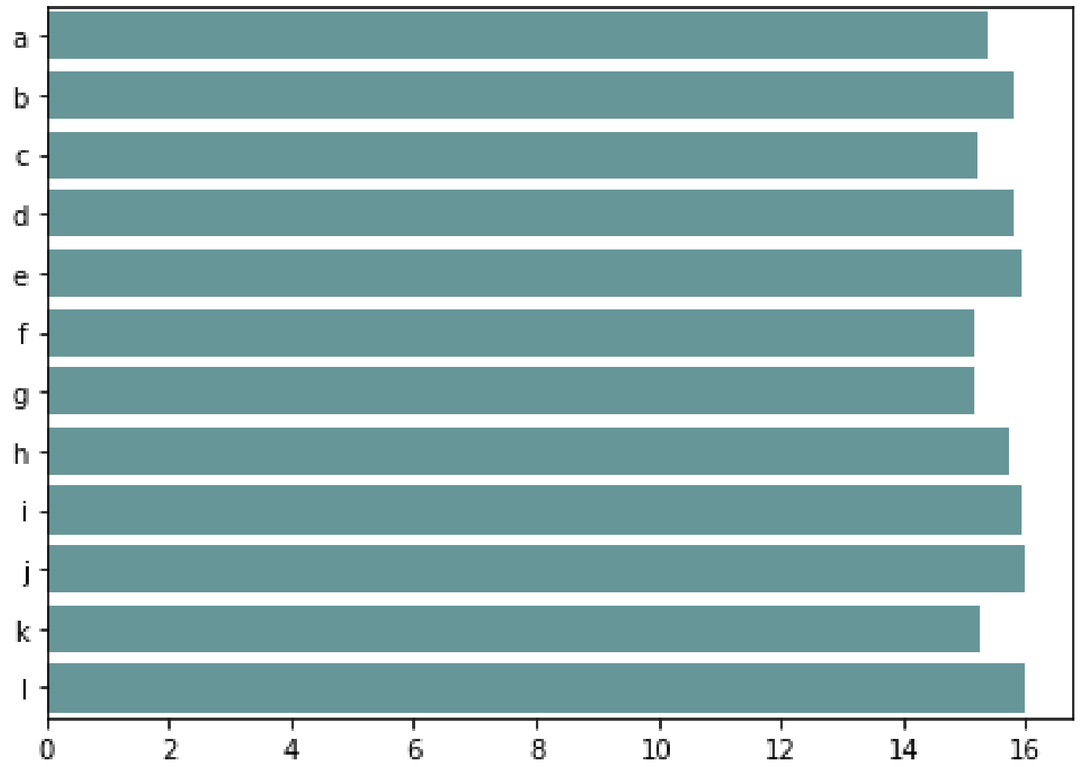



```
sns.barplot(x = values, y = ids,
            edgecolor = 'black')
```



Another solution

```
sns.barplot(x = values, y = ids,
            color = 'cadetblue')
```



IMPROVING YOUR DATA VISUALIZATIONS IN PYTHON

Let's paint some data!

IMPROVING YOUR DATA VISUALIZATIONS IN PYTHON

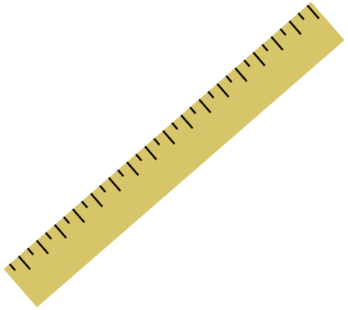
Continuous color palettes

Nick Strayer
Instructor

Continuous

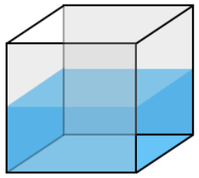
- *Ordered*
- *Lots of possible values*

Distances



Sensor Readings

Volumes

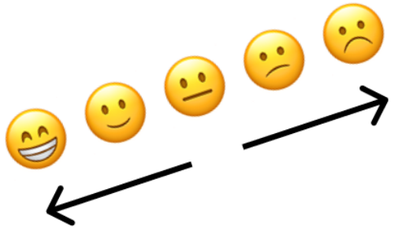


Not Continuous

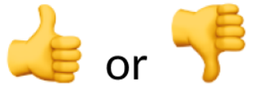
- *No order or...*
- *Few possible values*

Categories

Relative Scales

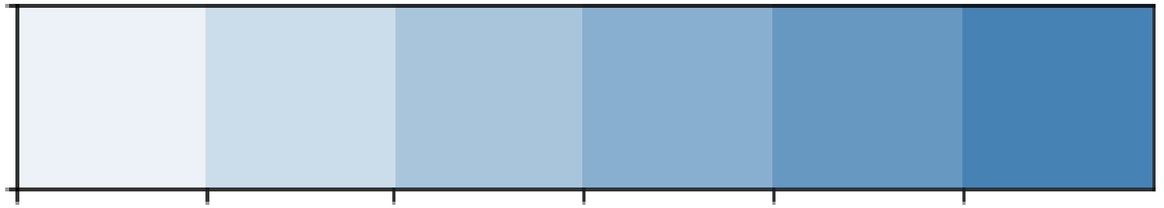


Binary Values

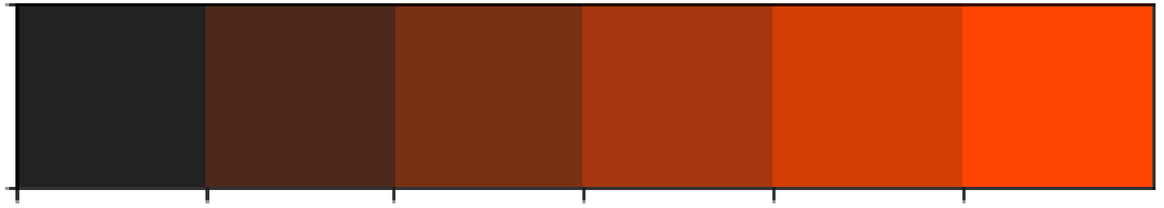


Seaborn's continuous palettes

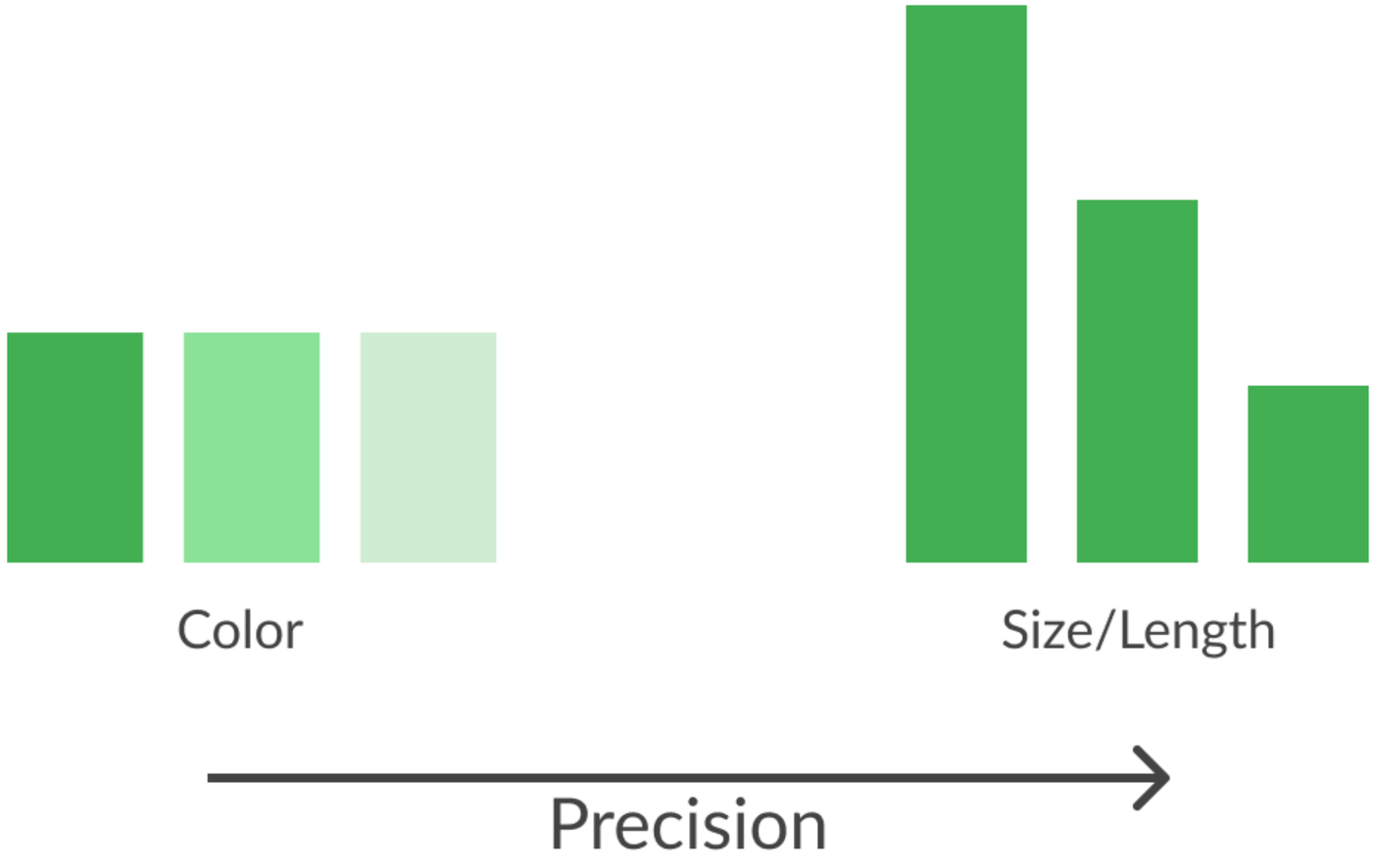
```
blue_scale = sns.light_palette("steelblue")
sns.palplot(blue_scale)
```



```
red_scale = sns.dark_palette("orangered")
sns.palplot(red_scale)
```

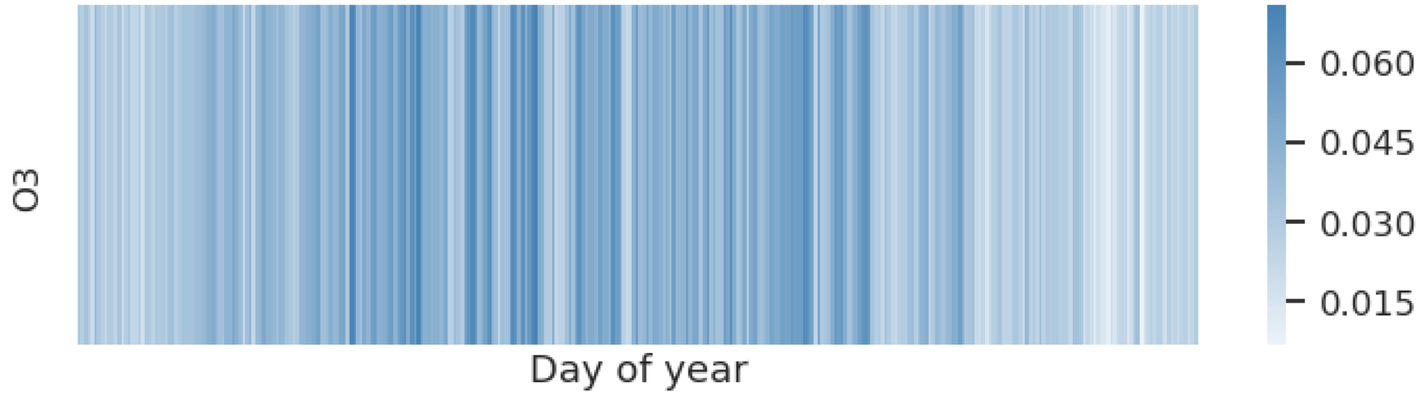


Color is less precise



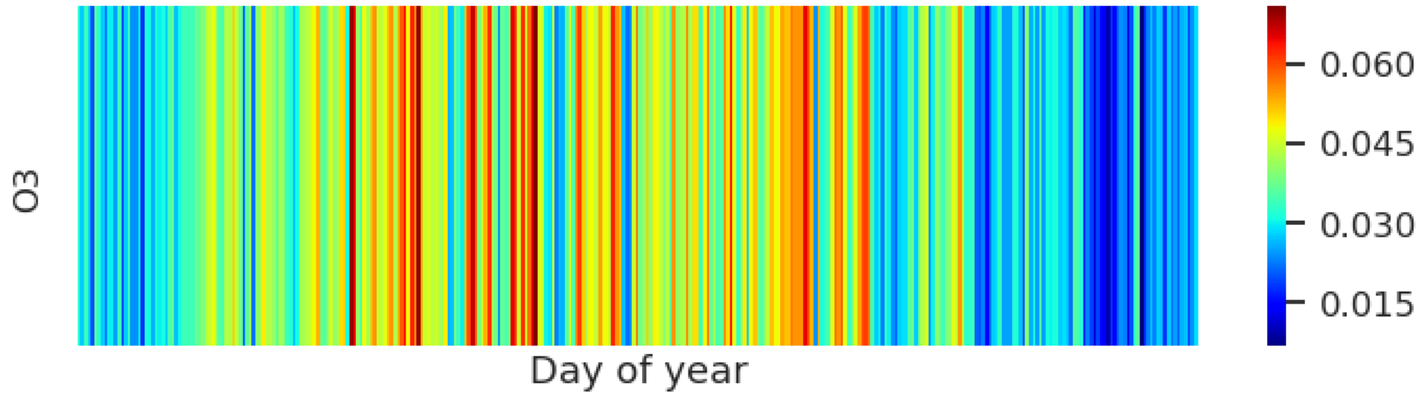
Keep it simple

```
indy_oct = pollution.query("year == 2015 & city == 'Indianapolis'")
blue_scale = sns.light_palette("steelblue", as_cmap = True)
sns.heatmap(indy_oct[['O3']], cmap = blue_scale)
```



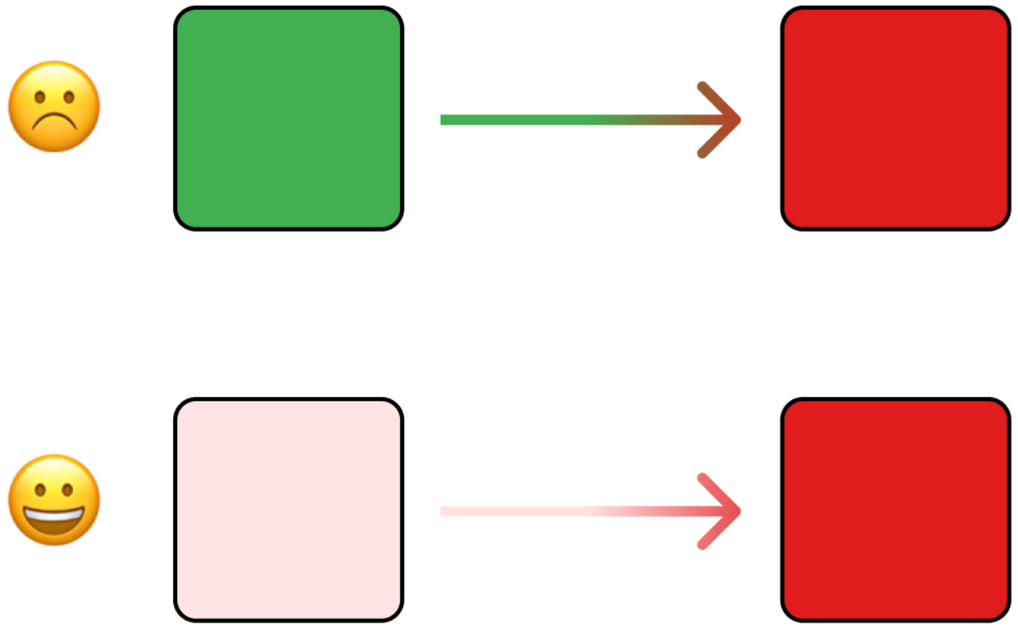
Keep it simple

```
indy_oct = pollution.query("year == 2015 & city == 'Indianapolis'")
jet_scale = palette = sns.color_palette('jet', as_cmap = True)
sns.heatmap(indy_oct[['O3']], cmap = jet_scale)
```



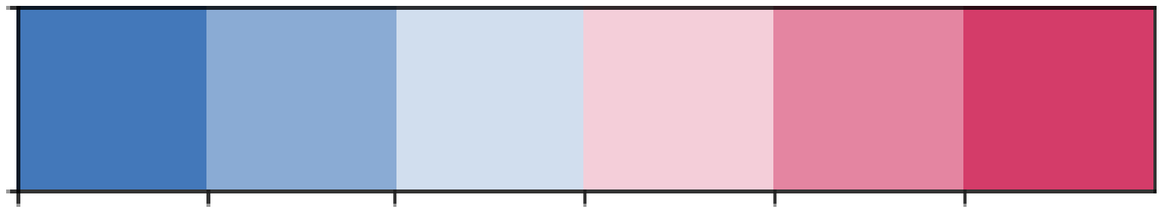
Be aware of color blindness

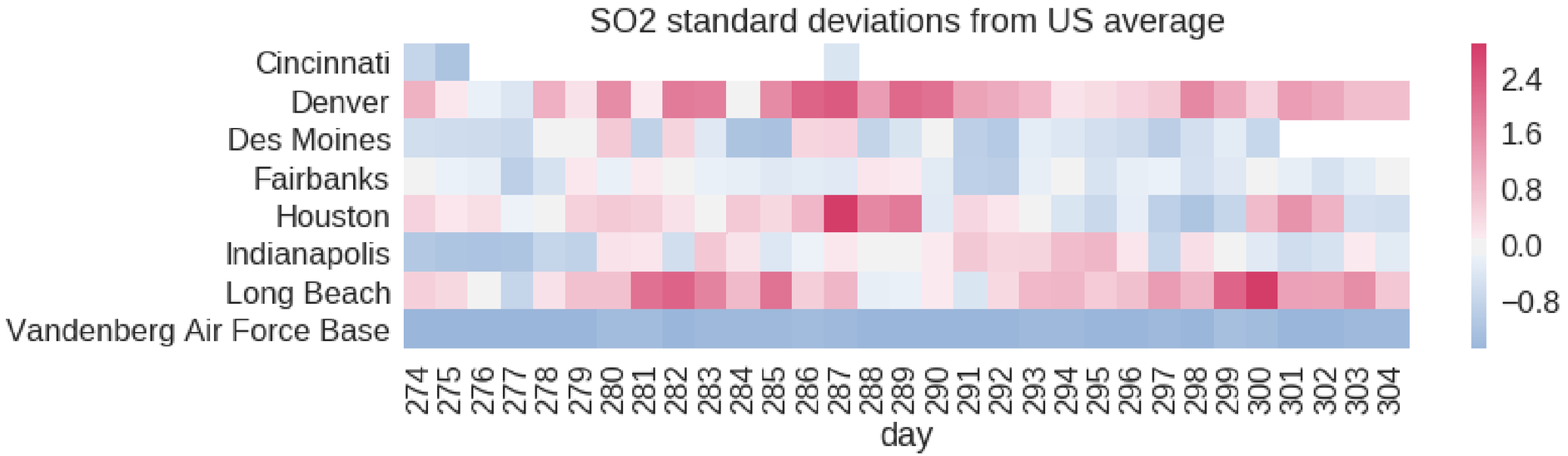
- Avoid transitions between green and red
- Palettes that use intensity are safer



Encoding neutral values

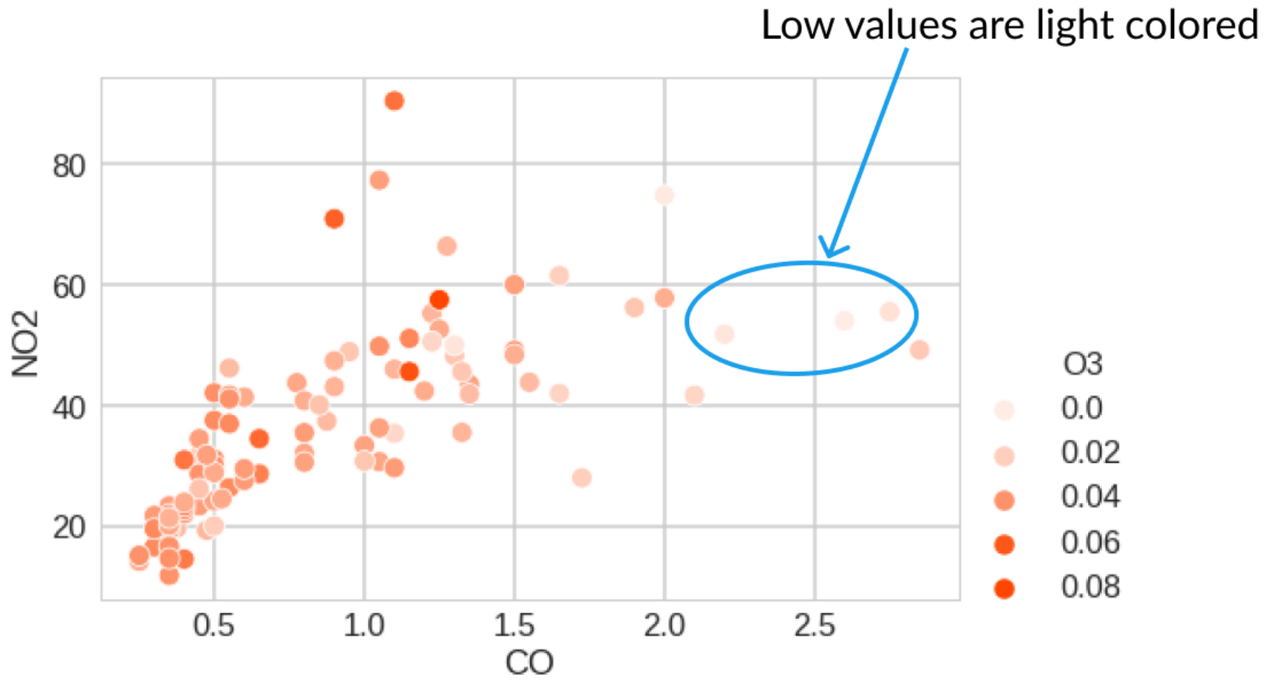
```
pal_light = sns.diverging_palette(250, 0)
pal_dark  = sns.diverging_palette(250, 0, center = 'dark')
```





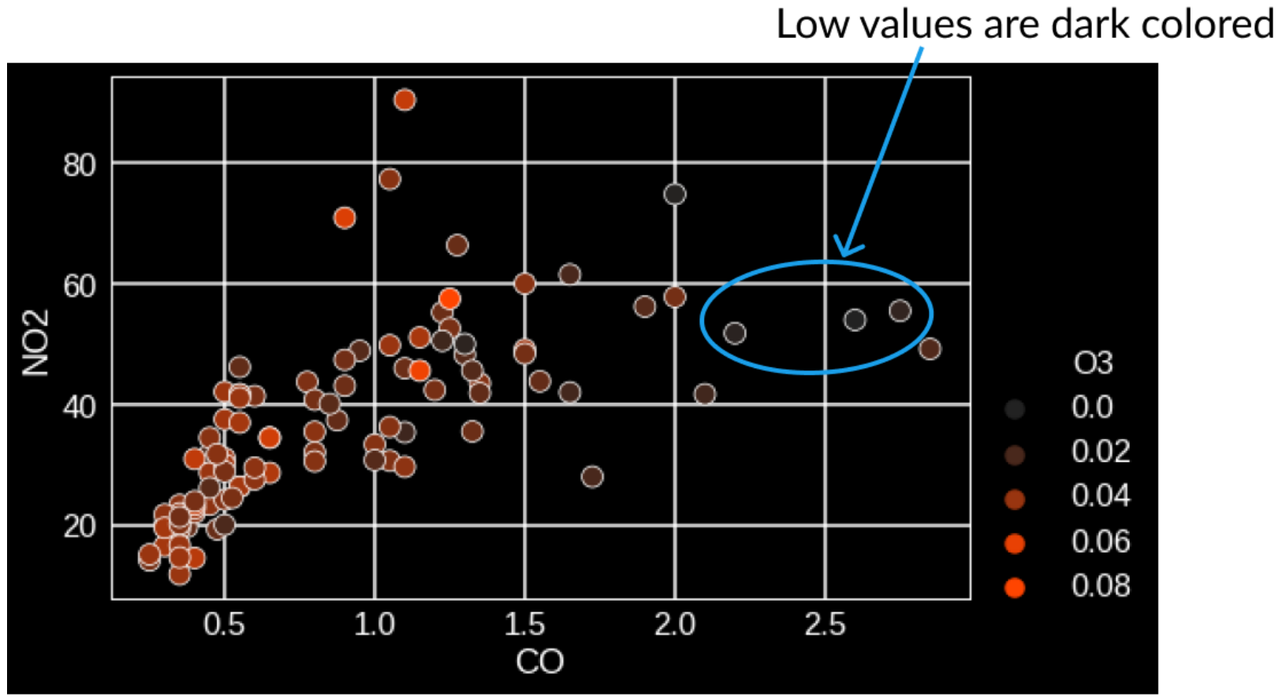
Be aware of contexts

```
plt.style.use('seaborn-white')

light_palette = sns.light_palette("orangered")
sns.scatterplot(x = 'CO', y = 'NO2', hue = 'O3', data = lb_2012,
                palette = light_palette)
```



Be aware of contexts

```
plt.style.use('dark_background')

dark_palette = sns.dark_palette("orangered")
sns.scatterplot(x = 'CO', y = 'NO2', hue = 'O3', data = lb_2012,
                palette = dark_palette)
```



IMPROVING YOUR DATA VISUALIZATIONS IN PYTHON

Let's continue in the exercises

IMPROVING YOUR DATA VISUALIZATIONS IN PYTHON

Categorical palettes

Nick Strayer
Instructor

Categorical Data

Cities

Cincinnati
Denver
Des Moines
Fairbanks
Houston
Indianapolis
Long Beach
Vandenberg Air Force Base

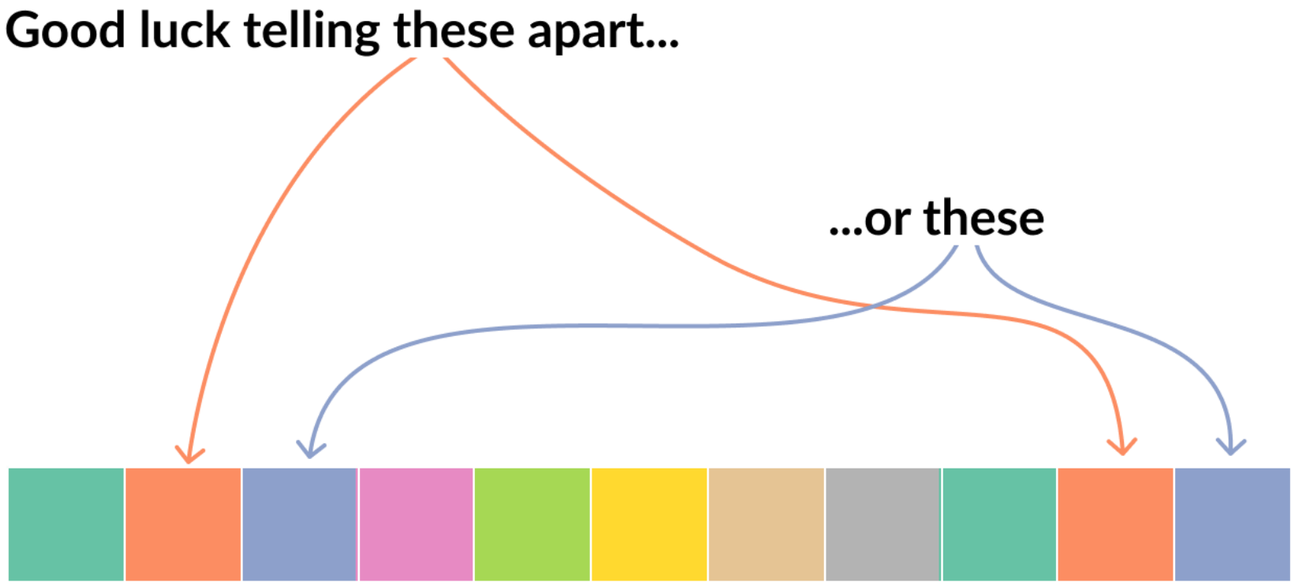
Countries

Bird Species

Limits in perception

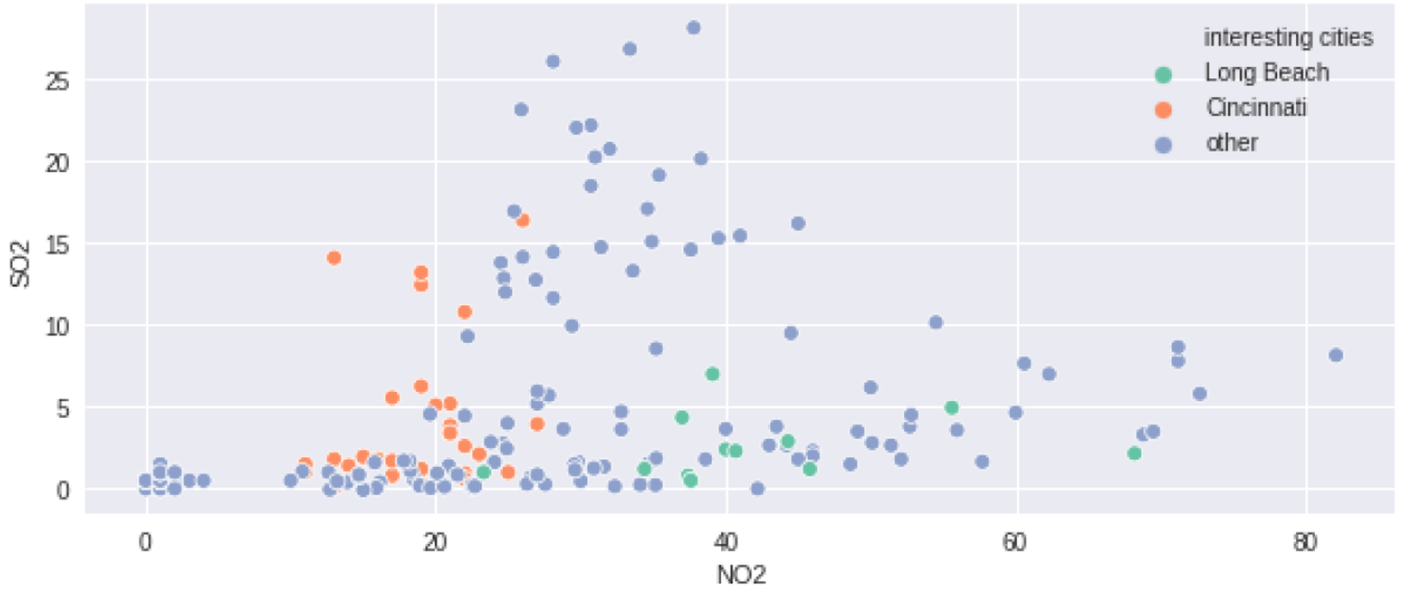
- Try and limit to 10 or fewer categories
- Keep color-blindness in mind

```
sns.palplot(sns.color_palette('Set2', 11))
```



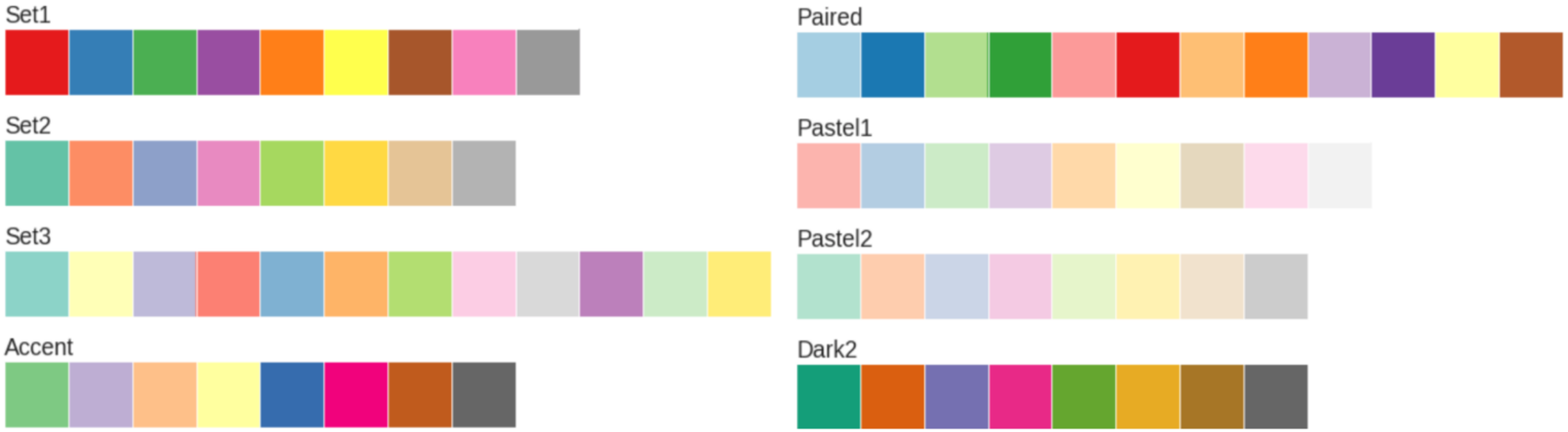
Dealing with these limitations

```
# Assign a new column to dataframe the desired combos
pollution['interesting cities'] = [x if x in ['Long Beach', 'Cincinnati']
                                   else 'other' for x in pollution['city'] ]

sns.scatterplot(x="NO2", y="SO2", hue = 'interesting cities', palette='Set2',
                data=pollution.query('year == 2014 & month == 12'))
```



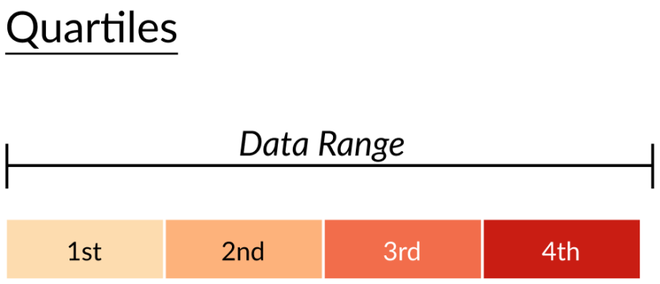
Seaborn's categorical palettes

```
colorbrewer_palettes = ['Set1',   'Set2',    'Set3',    'Accent',
                        'Paired', 'Pastel1', 'Pastel2', 'Dark2']

for pal in colorbrewer_palettes:
    sns.palplot(pal=sns.color_palette(pal))
    plt.title(pal, loc = 'left')
```



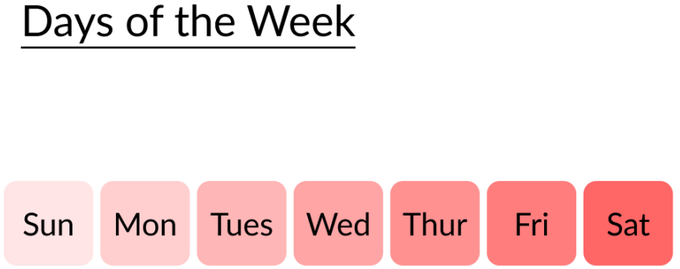
Ordinal data (a)

- Has order between classes
- A set number of distinct classes



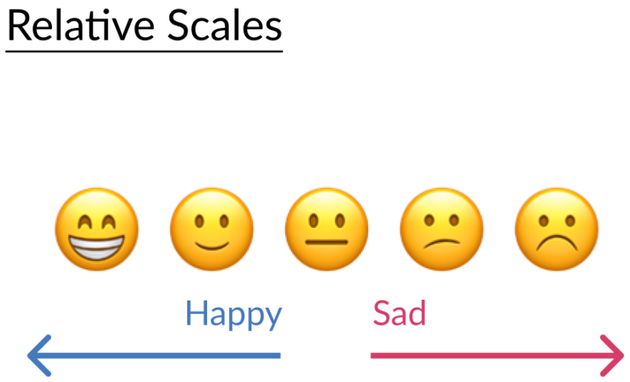
Ordinal data (b)

- Has order between classes
- A set number of distinct classes



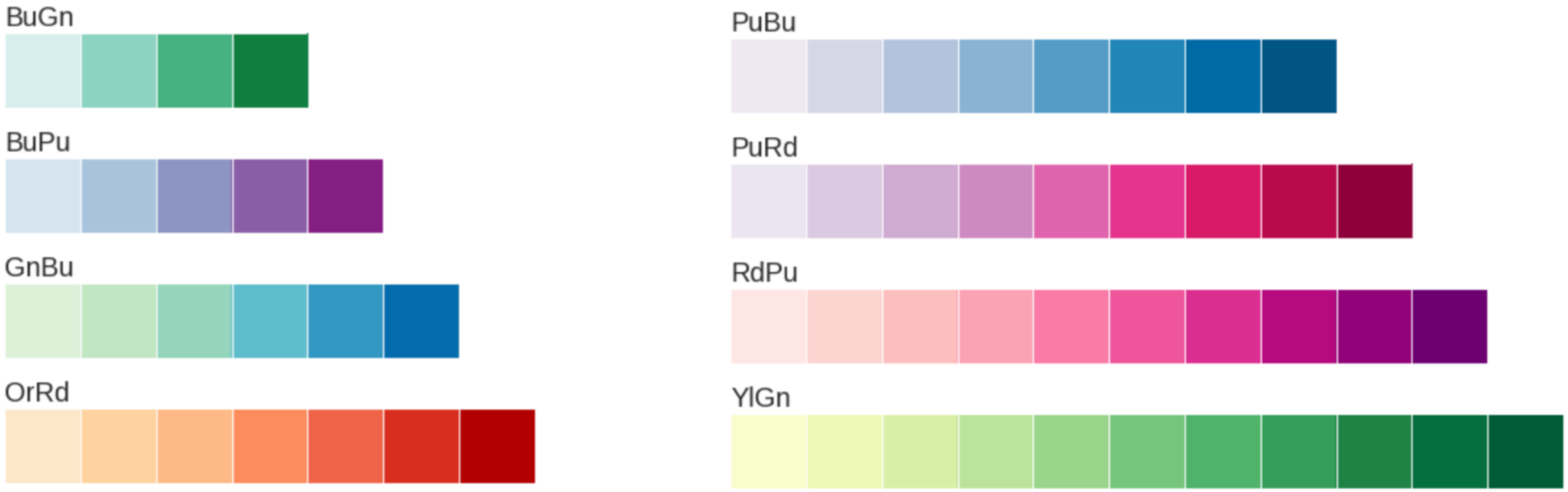
Ordinal data (c)

- Has order between classes
- A set number of distinct classes



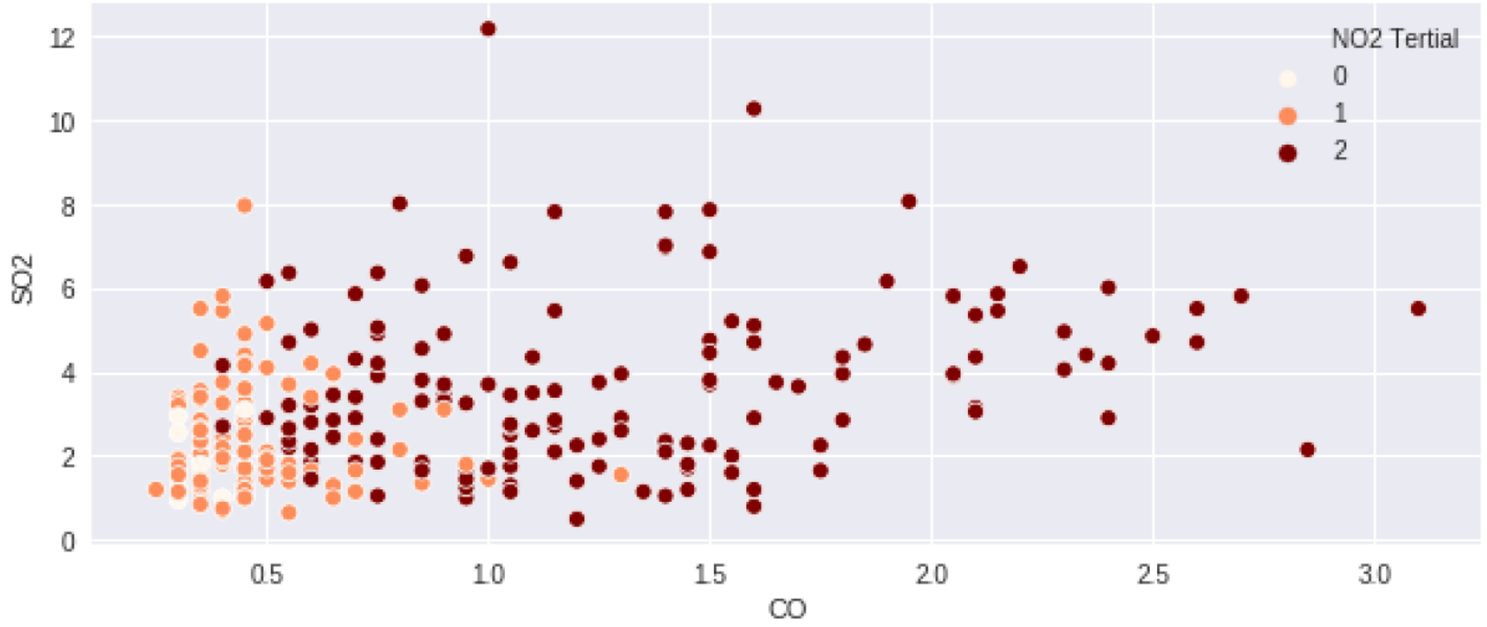
Building ordinal palettes

```
colorbrewer_palettes = ['Reds', 'Blues', 'YlOrBr', 'PuBuGn', 'GnBu', 'Greys']

for i, pal in enumerate(colorbrewer_palettes):
    sns.palplot(pal=sns.color_palette(pal, n_colors=i+4))
```



Palette shortcuts

```
# Make a tertials column using qcut()
pollution['NO2 Tertial'] = pd.qcut(pollution['NO2'], 3, labels = False)

# Plot colored by the computer tertials
sns.scatterplot(x="CO", y="SO2", hue='NO2 Tertial', palette="OrRd",
                data=pollution.query("city == 'Long Beach' & year == 2014"))
```



IMPROVING YOUR DATA VISUALIZATIONS IN PYTHON

Let's color some categories