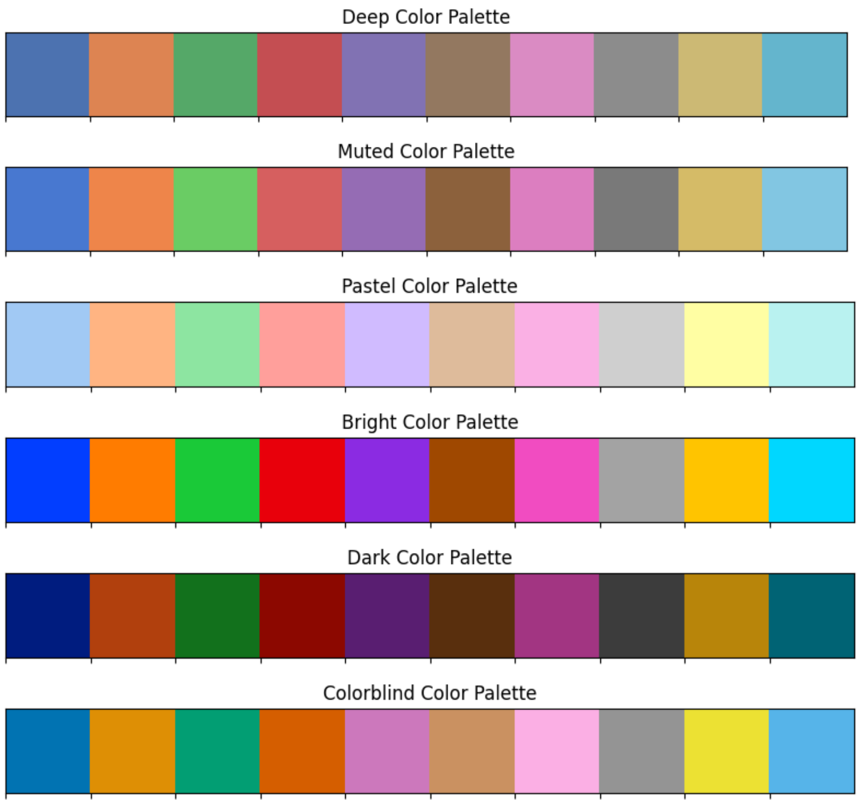
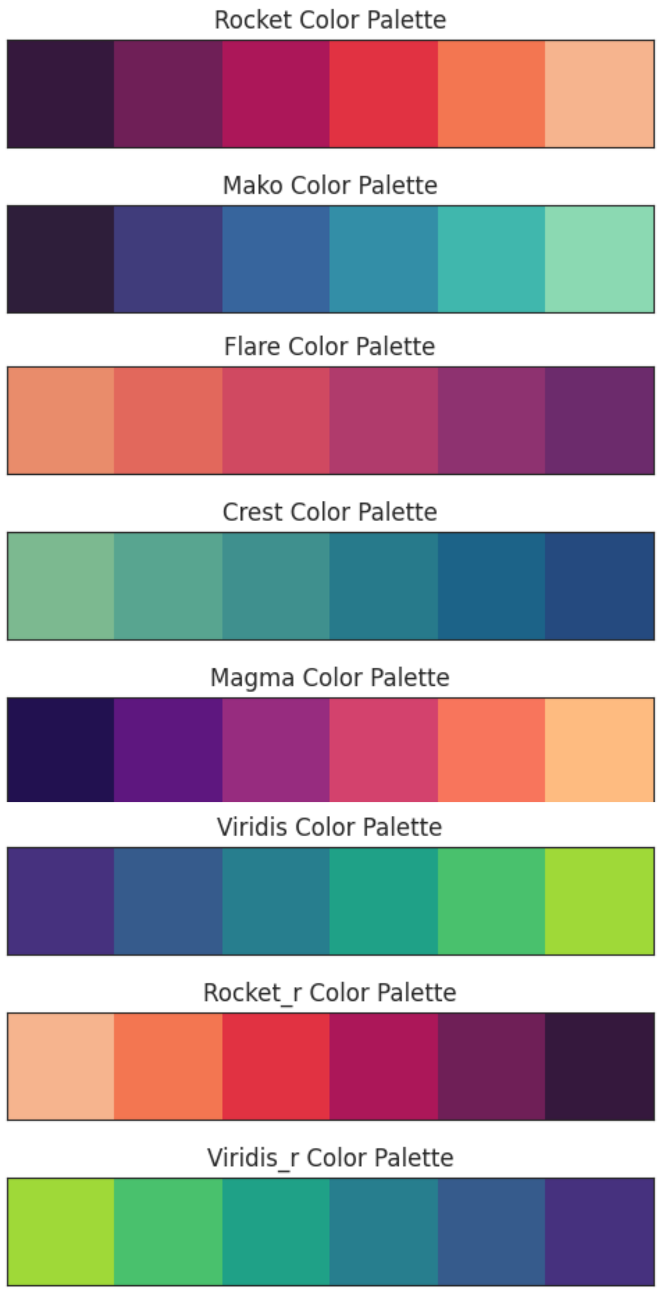
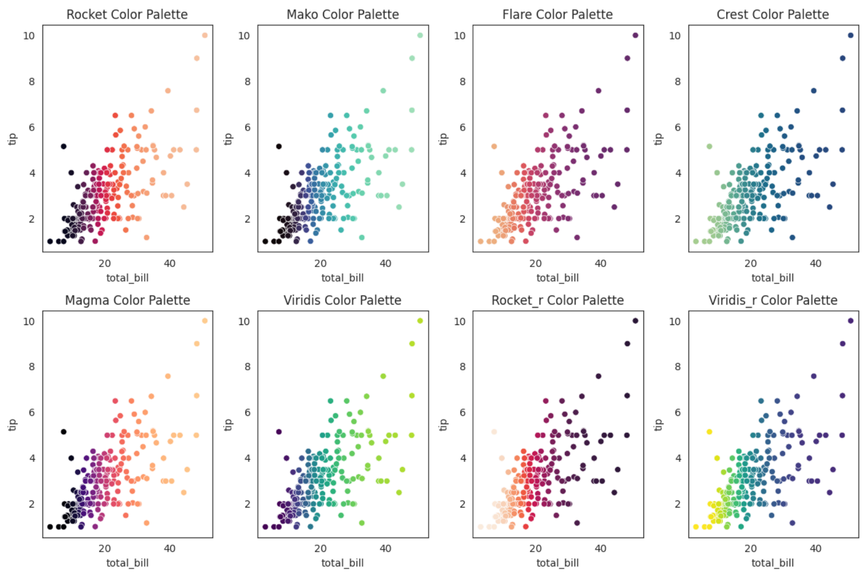
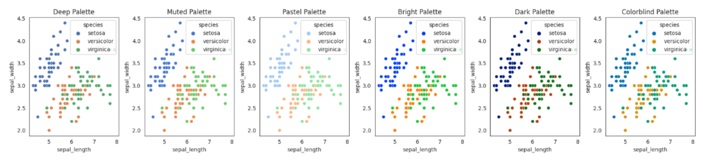
Seaborn Color Palettes

Seaborn simplifies the process of selecting colors that are well-suited for your data and visualization objectives. In this chapter, we'll explore the key principles that should guide your color choices and discover the useful tools provided by seaborn to assist you in finding the optimal color palette for your specific application. By leveraging seaborn's capabilities, you can easily enhance the visual representation of your data and create impactful visualizations.









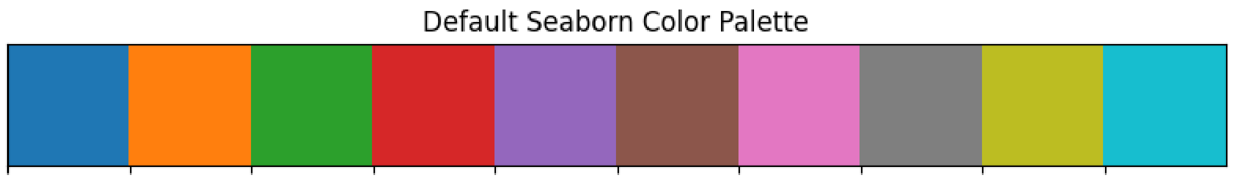
```python
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
# Ignore warnings
warnings.filterwarnings("ignore")
```

Qualitative color palettes

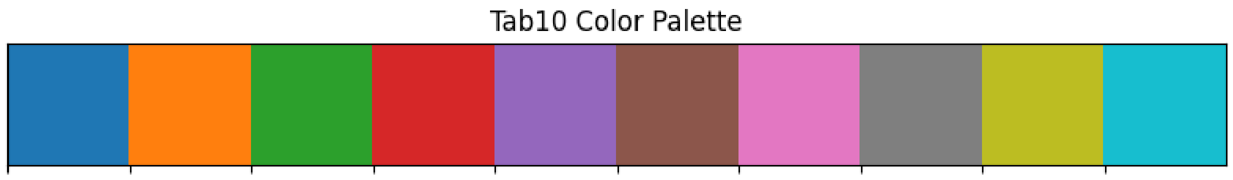
Qualitative palettes are well-suited to representing categorical data because most of their variation is in the hue component.

The default color palette in seaborn is a qualitative palette with ten distinct hues:

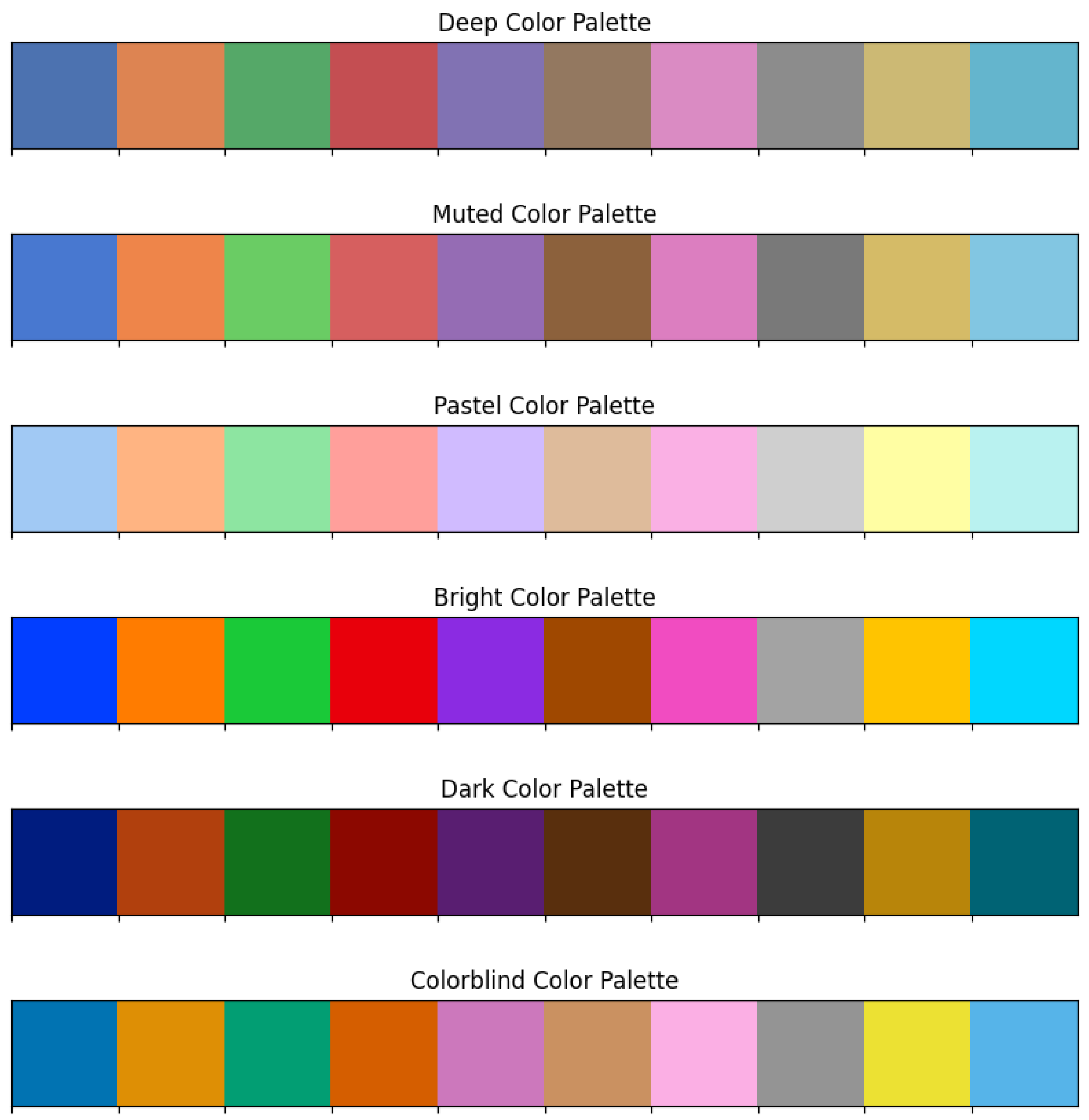
```python
# Default Seaborn color palette
default_palette = sns.color_palette()
sns.palplot(default_palette)
plt.title('Default Seaborn Color Palette')
plt.show()
```



```python
# Tab10 color palette
tab10_palette = sns.color_palette("tab10")
sns.palplot(tab10_palette)
plt.title('Tab10 Color Palette')
plt.show()
```

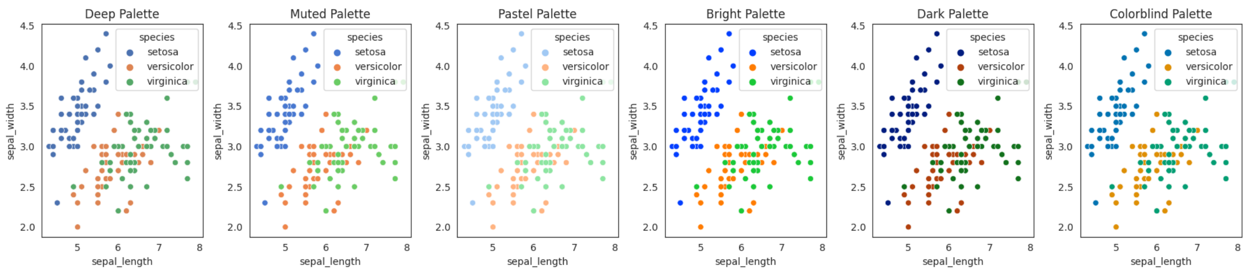


```python
# Qualitative color palettes in Seaborn
qualitative_palettes = ['deep', 'muted', 'pastel', 'bright', 'dark',
'colorblind']
for palette_name in qualitative_palettes:
    palette = sns.color_palette(palette_name)
    sns.palplot(palette)
    plt.title(f'{palette_name.capitalize()} Color Palette')
    plt.show()
```

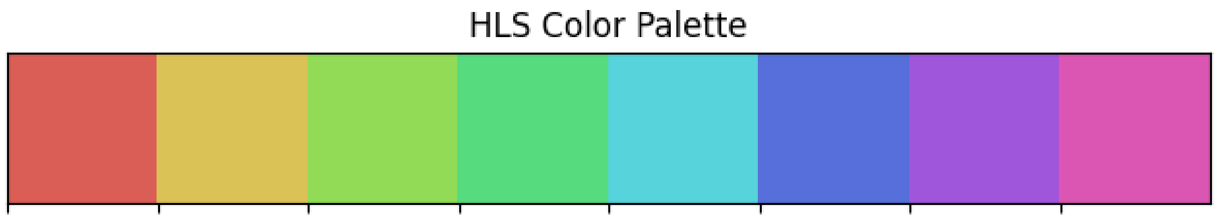


```
import seaborn as sns
import matplotlib.pyplot as plt

# Qualitative color palettes in Seaborn
qualitative_palettes = ['deep', 'muted', 'pastel', 'bright', 'dark',
'colorblind']
sample_data = sns.load_dataset('iris')

plt.figure(figsize=(18, 4))
sns.set_style('white')
```

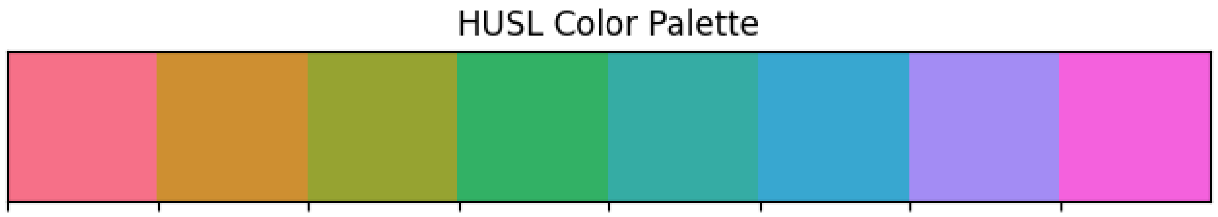
```
for i, palette_name in enumerate(qualitative_palettes):
    palette = sns.color_palette(palette_name)
    plt.subplot(1, 6, i + 1)
    sns.scatterplot(data=sample_data, x='sepal_length',
y='sepal_width', hue='species', palette=palette)
    plt.title(f'{palette_name.capitalize()} Palette')

plt.tight_layout()
plt.show()
```



```
# Circular color systems - hls and husl color palettes
hls_palette = sns.color_palette("hls", 8)
sns.palplot(hls_palette)
plt.title('HLS Color Palette')
plt.show()
```

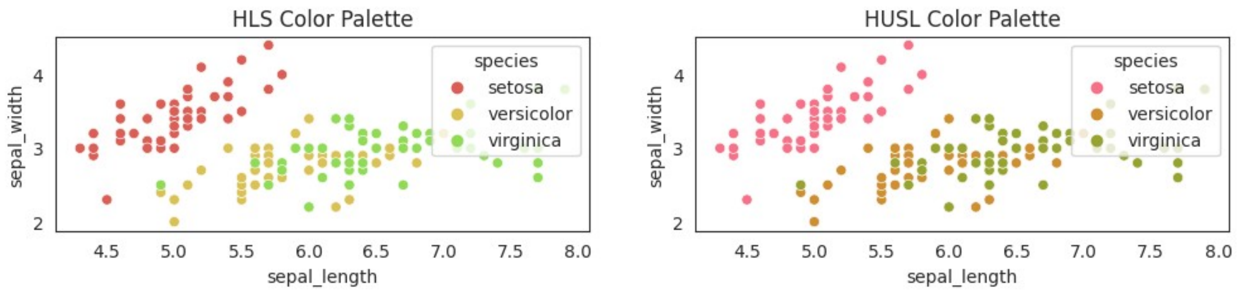


```
husl_palette = sns.color_palette("husl", 8)
sns.palplot(husl_palette)
plt.title('HUSL Color Palette')
plt.show()
```

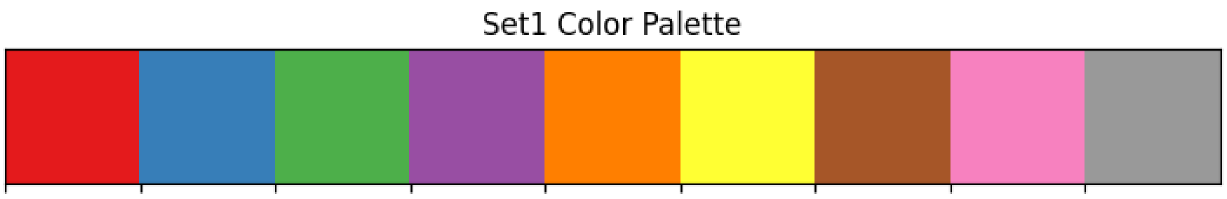


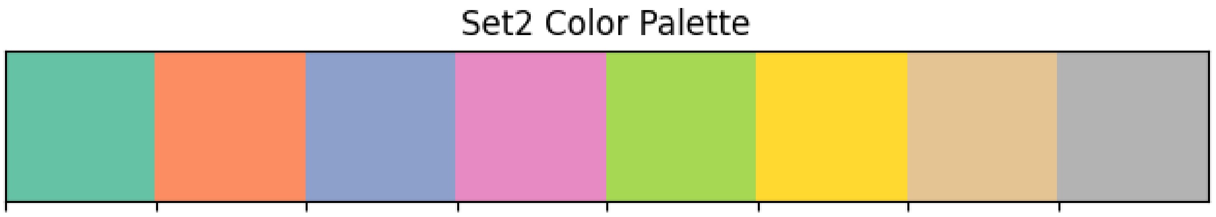
```
# Circular color systems - hls and husl color palettes
hls_palette = sns.color_palette("hls", 8)
```

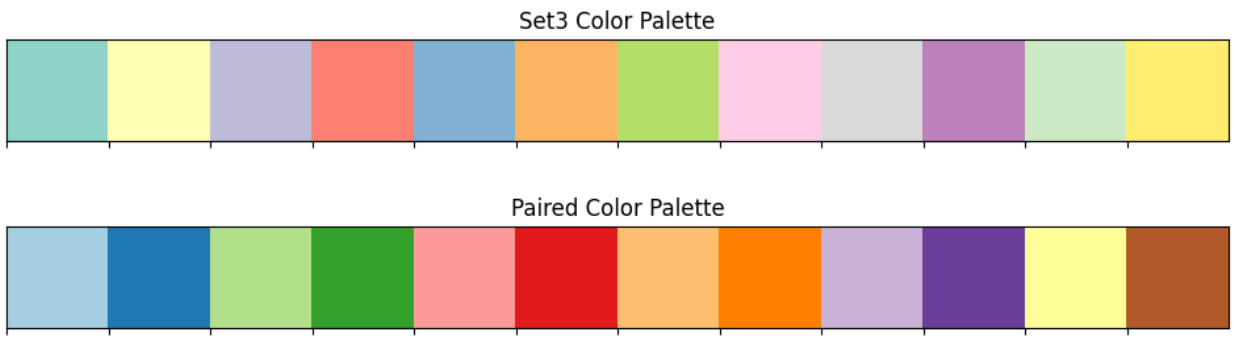
```
husl_palette = sns.color_palette("husl", 8)
plt.figure(figsize=(12, 2))
sns.set_style('white')
plt.subplot(1, 2, 1)
sns.scatterplot(data=sample_data, x='sepal_length', y='sepal_width',
hue='species', palette=hls_palette)
plt.title('HLS Color Palette')
plt.subplot(1, 2, 2)
sns.scatterplot(data=sample_data, x='sepal_length', y='sepal_width',
hue='species', palette=husl_palette)
plt.title('HUSL Color Palette')
plt.show()
```



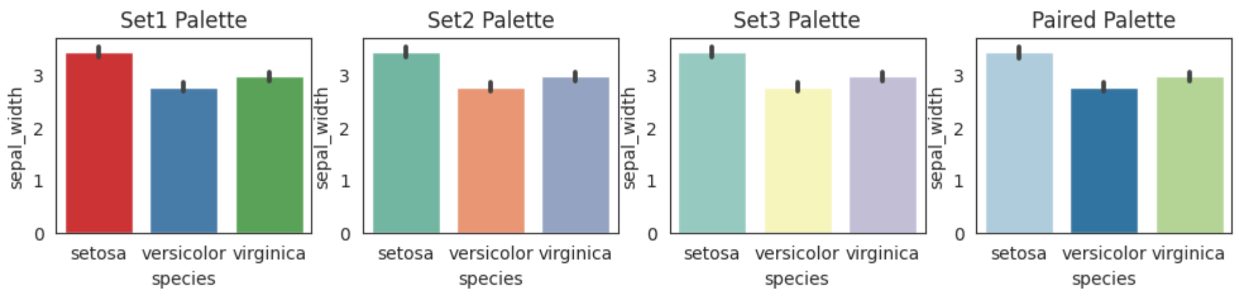
```
# Categorical Color Brewer palettes
categorical_palettes = ['Set1', 'Set2', 'Set3', 'Paired']
for palette_name in categorical_palettes:
    palette = sns.color_palette(palette_name)
    sns.palplot(palette)
    plt.title(f'{palette_name.capitalize()} Color Palette')
    plt.show()
```







```
# Categorical Color Brewer palettes
categorical_palettes = ['Set1', 'Set2', 'Set3', 'Paired']
plt.figure(figsize=(12, 2))
sns.set_style('white')
for i, palette_name in enumerate(categorical_palettes):
    palette = sns.color_palette(palette_name)
    plt.subplot(1, len(categorical_palettes), i + 1)
    sns.barplot(data=sample_data, x='species', y='sepal_width',
palette=palette)
    plt.title(f'{palette_name.capitalize()} Palette')
plt.show()
```

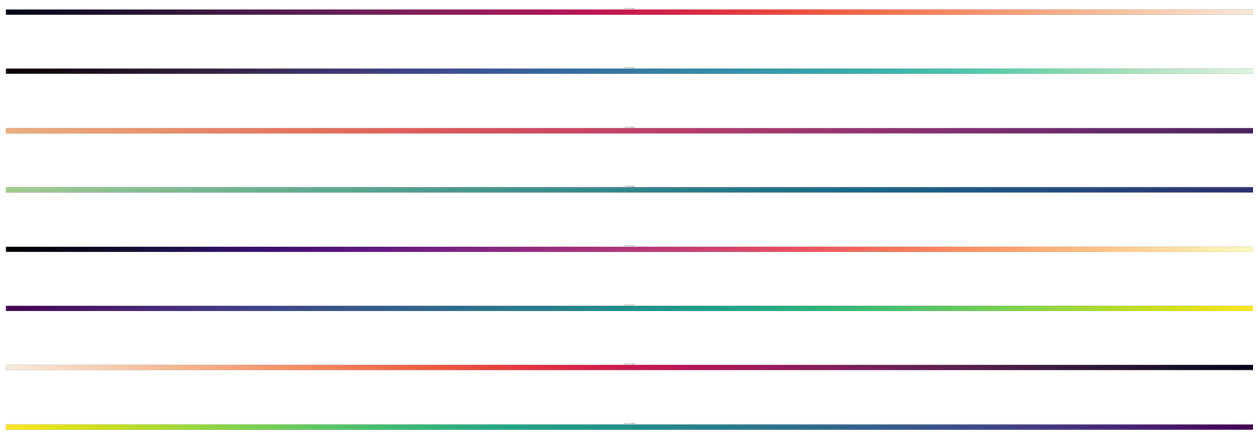


Sequential color palettes

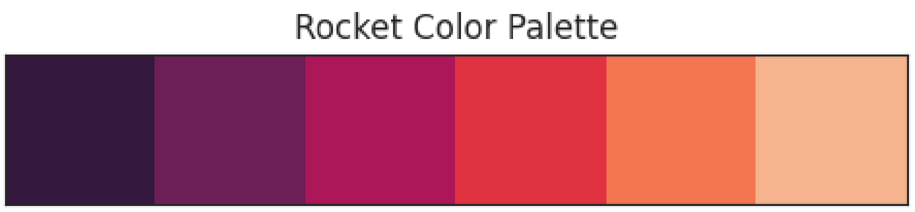
The second major class of color palettes is called "sequential".

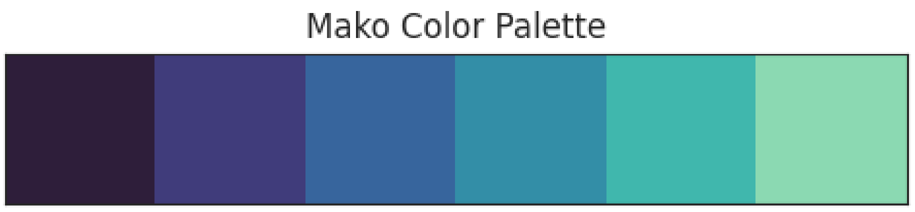
This kind of mapping is appropriate when data range from relatively low or uninteresting values to relatively high or interesting values (or vice versa).

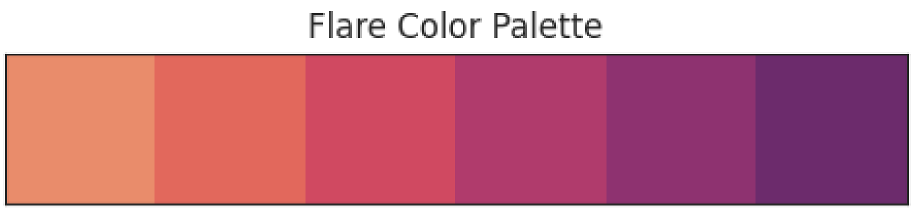
```
# Perceptually uniform palettes
perceptually_uniform_palettes = ['rocket', 'mako', 'flare', 'crest',
'magma', 'viridis', 'rocket_r', 'viridis_r']

for palette_name in perceptually_uniform_palettes:
    cmap = sns.color_palette(palette_name, as_cmap=True)
    sns.palplot(cmap.colors)
    plt.title(f'{palette_name.capitalize()} Color Palette')
    plt.show()
```

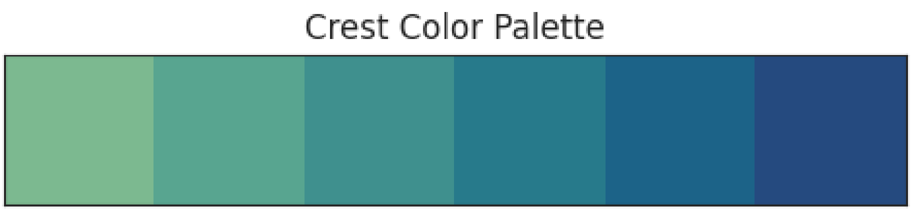


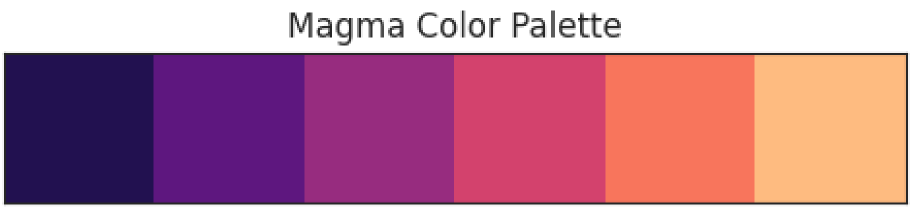
```
# Perceptually uniform palettes - Discrete
Perceptually_uniform_palettes = ['rocket', 'mako', 'flare',
'crest','magma','viridis','rocket_r','viridis_r']
for palette_name in Perceptually_uniform_palettes:
    palette = sns.color_palette(palette_name)
    sns.palplot(palette)
    plt.title(f'{palette_name.capitalize()} Color Palette')
    plt.show()
```

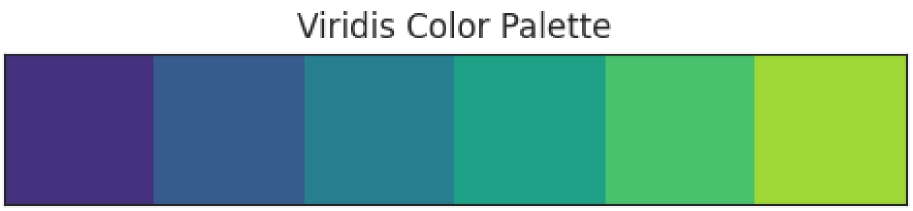


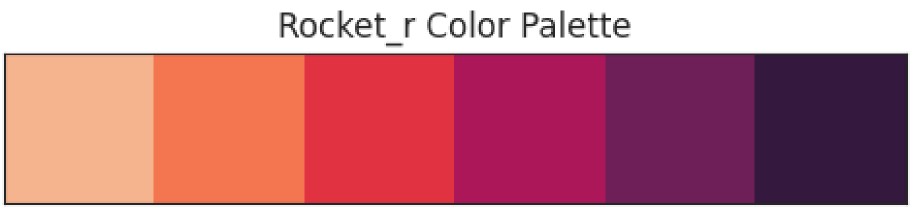


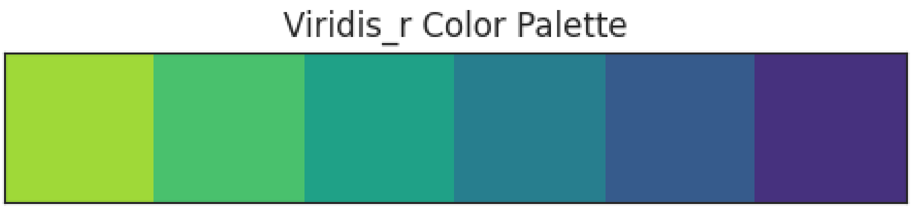




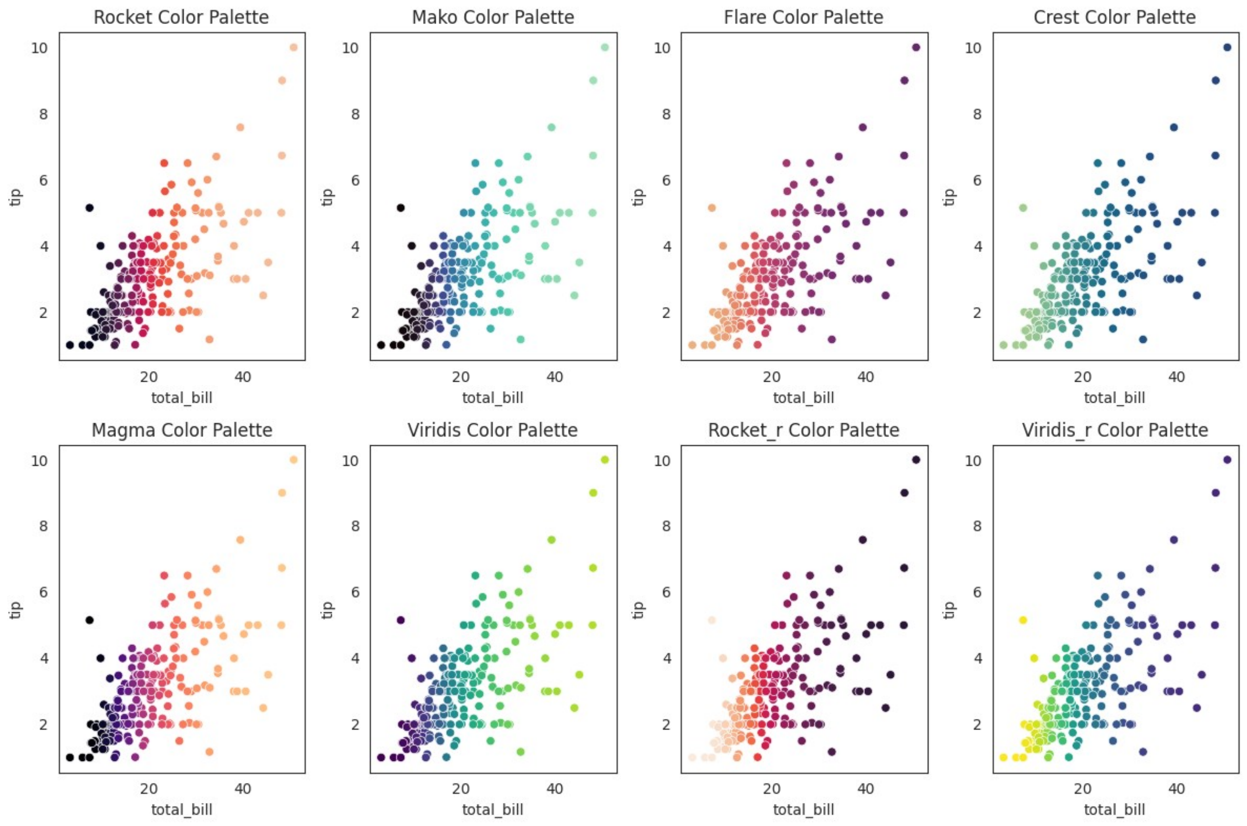






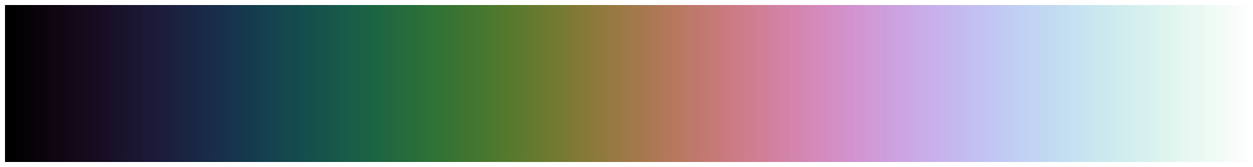


```
# Load sample data
tips = sns.load_dataset('tips')

# Define the perceptually uniform palettes
perceptually_uniform_palettes = ['rocket', 'mako', 'flare', 'crest',
'magma', 'viridis', 'rocket_r', 'viridis_r']

# Create the plot
plt.figure(figsize=(12, 8))

for i, palette_name in enumerate(perceptually_uniform_palettes):
    cmap = sns.color_palette(palette_name, as_cmap=True)
    colors = list(cmap.colors)  # Convert colormap to list of colors
    ax = plt.subplot(2, 4, i+1)
    sns.scatterplot(data=tips, x='total_bill', y='tip', 
hue='total_bill', palette=colors, legend=False)
    plt.title(f'{palette_name.capitalize()} Color Palette')

plt.tight_layout()
plt.show()
```

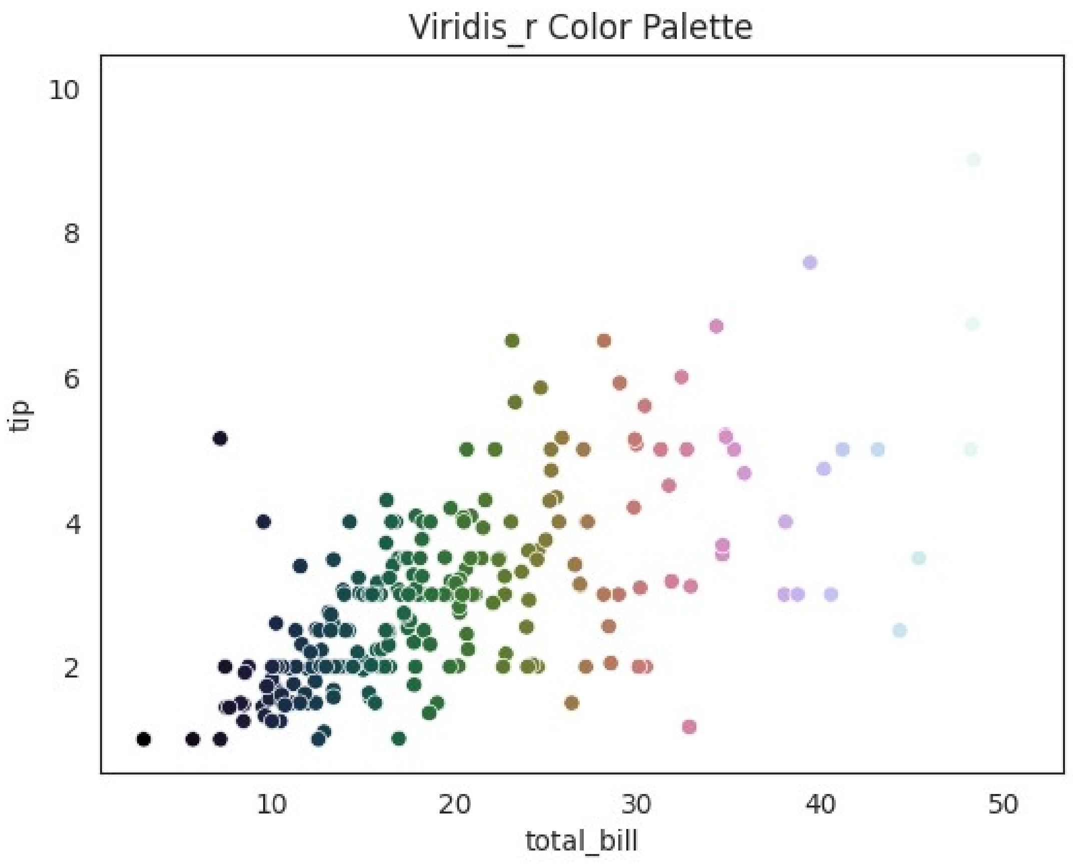


Sequential "cubehelix" palettes

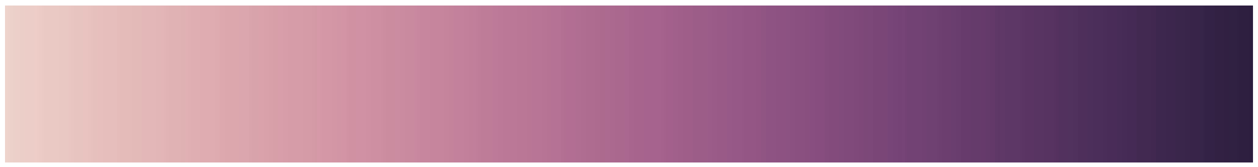
```
sns.color_palette("cubehelix", as_cmap=True)
```



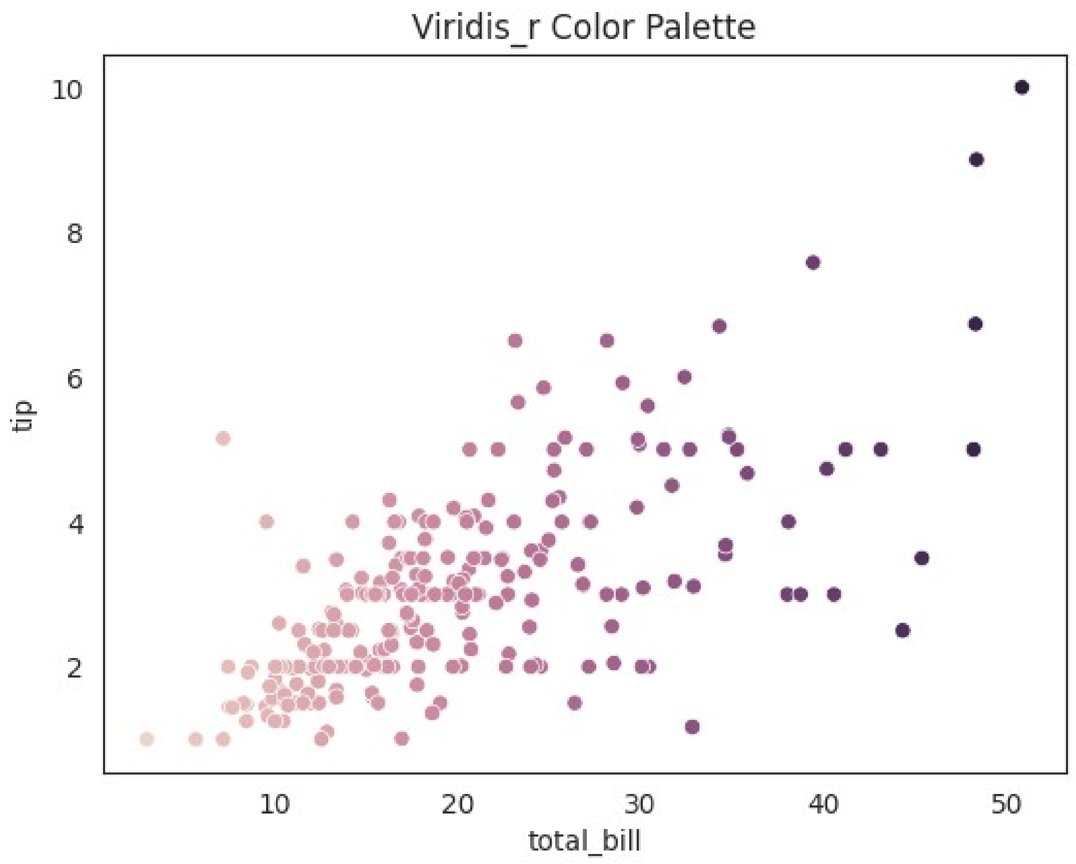
```
a= sns.color_palette("cubehelix", as_cmap=True)
sns.scatterplot(data=tips, x='total_bill', y='tip', hue='total_bill', 
palette=a, legend=False)
plt.title(f'{palette_name.capitalize()} Color Palette')
plt.show()
```



```
sns.cubehelix_palette(as_cmap=True)
```



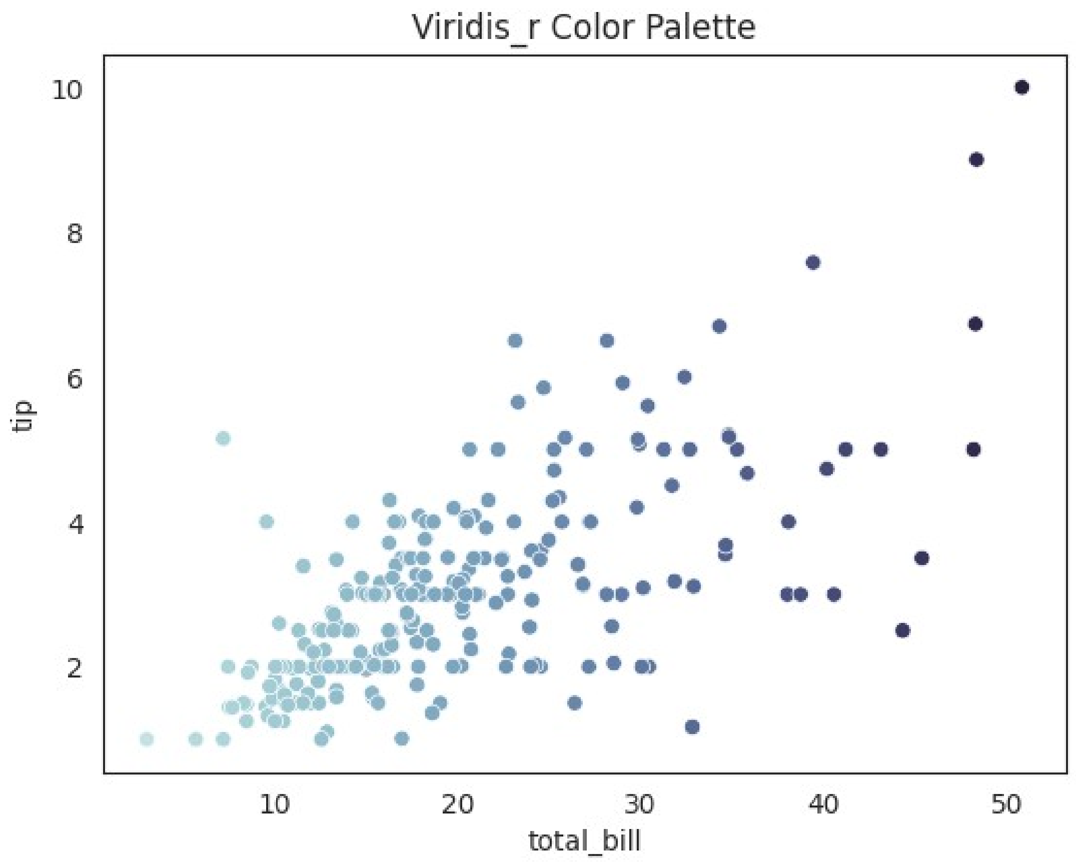
```
a= sns.cubehelix_palette(as_cmap=True)
sns.scatterplot(data=tips, x='total_bill', y='tip', hue='total_bill',
palette=a, legend=False)
plt.title(f'{palette_name.capitalize()} Color Palette')
plt.show()
```



```
sns.cubehelix_palette(start=.5, rot=-.5, as_cmap=True)
```

```
sns.color_palette("ch:start=.2,rot=-.3", as_cmap=True)
```

```
a= sns.color_palette("ch:start=.2,rot=-.3", as_cmap=True)
sns.scatterplot(data=tips, x='total_bill', y='tip', hue='total_bill',
palette=a, legend=False)
plt.title(f'{palette_name.capitalize()} Color Palette')
plt.show()
```



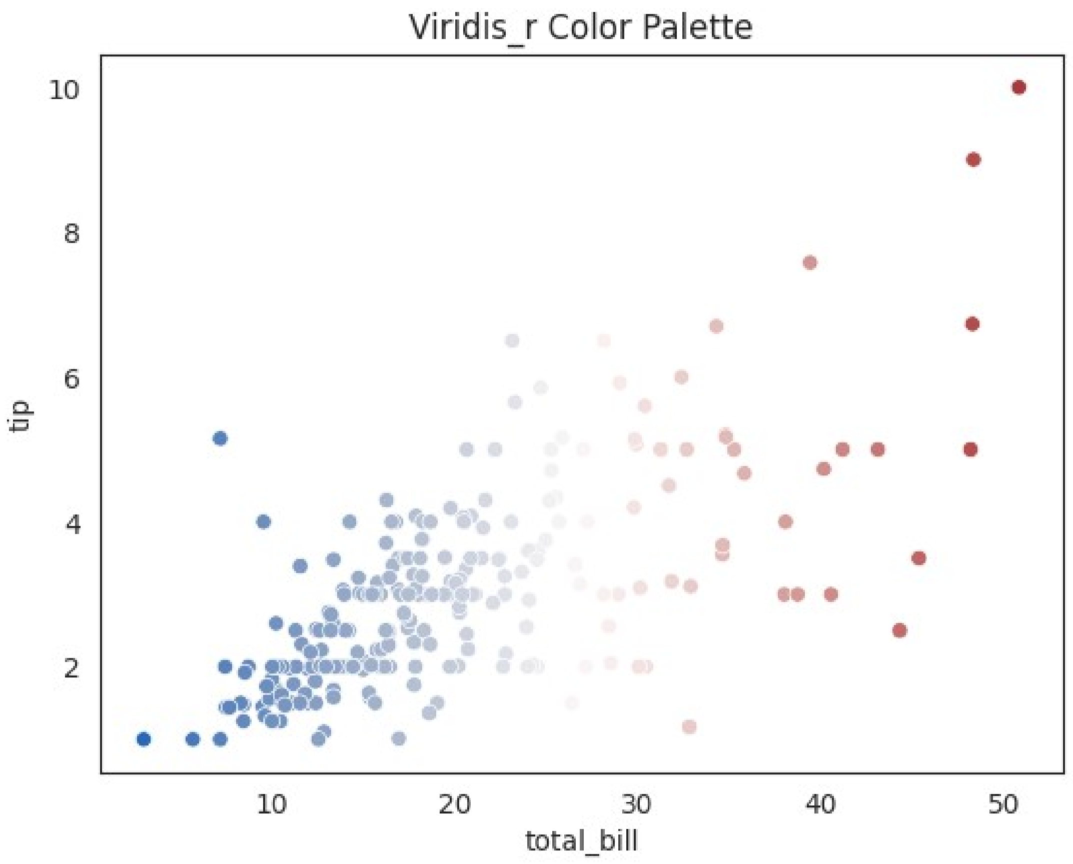
Diverging color palettes

The third class of color palettes is called “diverging”.

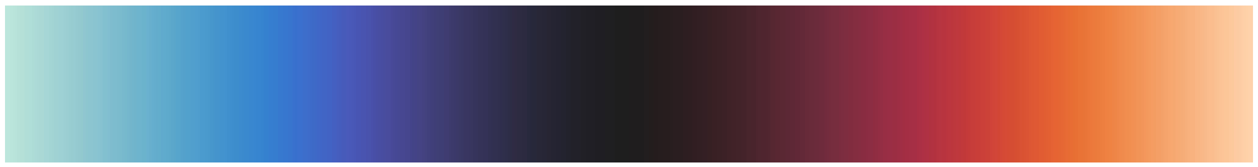
These are used for data where both large low and high values are interesting and span a midpoint value (often 0) that should be demphasized.

```
sns.color_palette("vlag", as_cmap=True)
```

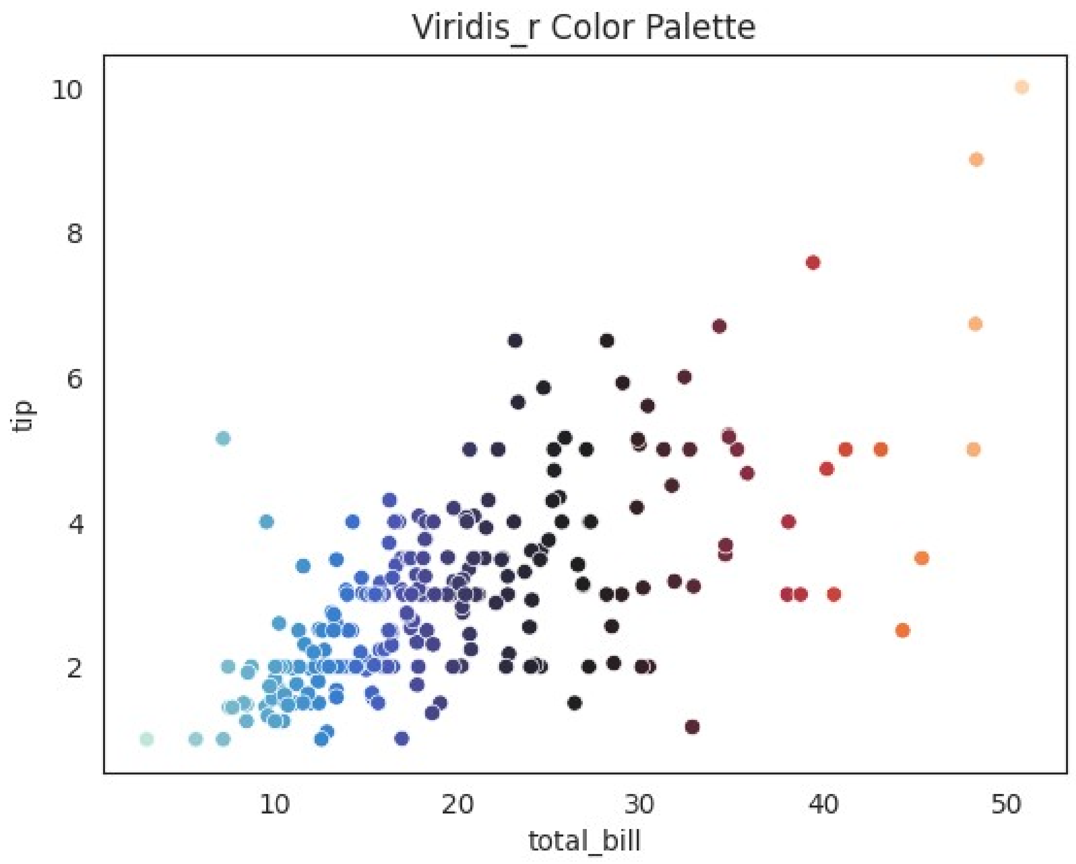
```
a= sns.color_palette("vlag", as_cmap=True)
sns.scatterplot(data=tips, x='total_bill', y='tip', hue='total_bill',
palette=a, legend=False)
plt.title(f'{palette_name.capitalize()} Color Palette')
plt.show()
```



```
sns.color_palette("icefire", as_cmap=True)
```



```
a= sns.color_palette("icefire", as_cmap=True)
sns.scatterplot(data=tips, x='total_bill', y='tip', hue='total_bill',
palette=a, legend=False)
plt.title(f'{palette_name.capitalize()} Color Palette')
plt.show()
```


Viridis_r Color Palette
10
8
6
4
2
tip
10
20
30
40
50
total_bill